第４章　データ転送システム

4.1 分散並列型動的解析用基本データの取得

本章では、分散並列型解析システムの中で、特に、動的ソルバーにおけるデータ転送システムについて解説する。ここでは、図 4-1 に示すワーレントラス型アーチ構造物の解析を通して、データ転送が正確に行われていることを説明する。

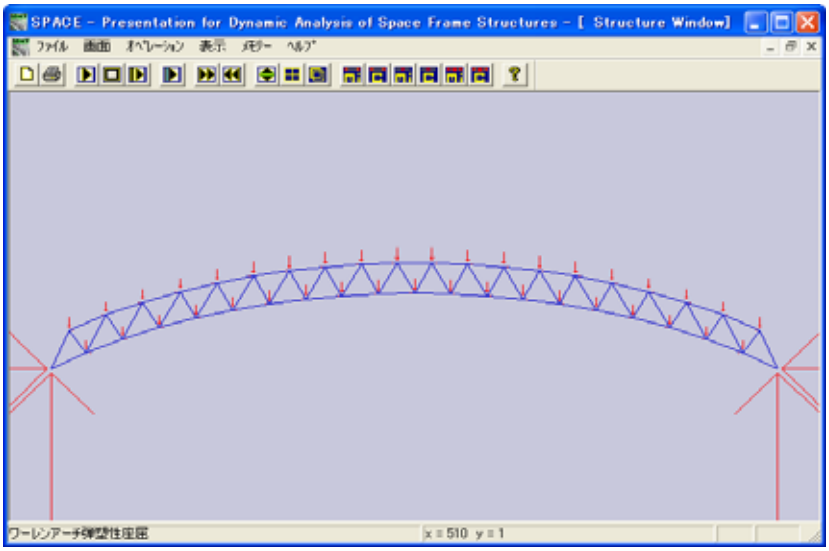

図 4-1　ワーレントラス型アーチ

分散並列環境を設定するファイルの中身は、以下のようであり、マスターとスレーブ 2 で PC クラスターを構成し、3 つの PC による分散並列型で動的解析を行うものとする。

```
sf3pa_slave.exe
Pcg-gr9f     1      27
Pcg-gr10f    28     53
Pcg-gr11f    54     79
```

マスターにおける分散並列環境が、上記のプロセス設定用ファイルを用いて、以下のように構築される。まず、マスター側の動的ソルバーで、解析システムよりコントロール情報を、以下に示すプログラムを用いて取得する。

```
c―――――― ★★　システムからのコントロール情報を取得
c――――――――――――――――――――――――――――――――――――――――――――――――――――★MPI 上での自己の状態を得る
! 解析プロセスの総数を取得
      call MPI_Comm_size(MPI_COMM_WORLD, n_proc, ierr)
! 自分のランクを取得
```

```
      call MPI_Comm_rank(MPI_COMM_WORLD, n_myRank, ierr)
      write(damp_out,'(a,2i5)')'(n_proc, myRank) =', n_proc, n_myRank
      ALLOCATE (nm_parallel( 2, 0:n_proc))
c-------------------------------------------------★担当部材リストを取得
      do i=0, n_proc-1
      call RequestMember(nm_parallel, i)
      enddo
      n_member1=nm_parallel(1, n_myRank)                    ! 担当部材の先頭番号
      n_member2=nm_parallel(2, n_myRank)                    ! 担当部材の最終番号
      max_member=n_member2-n_member1+1     ! 担当部材の最大数
      write(damp_out,'(a,i5,a,i5,a,i5)')'max_member=',max_member,
     +' n_member1=',n_member1,' n_member2=',n_member2
```

上記プログラムの出力は、以下のようである。

```
マスター:        (n_proc, myRank) =    3    0
                 rank:   0  担当部材:         1   -        27
                 rank:   1  担当部材:        28   -        53
                 rank:   2  担当部材:        54   -        79
                 max_member=   27 n_member1=    1 n_member2=   27
```

これを受けて、２つのスレーブ側では、次に示すサブルーチンを用いて、並列環境を整える。

```
c------- ★★解析制御情報をマスターから取得
c-------------------------------------------------★並列処理用予備計算（処理）（前）
c-------------------------------------------------★MPI 上での自己の状態を得る
c                                    自分のランクを取得
      call MPI_Comm_rank(MPI_COMM_WORLD, n_myRank, nErr)
c                                   プロセスの総数を取得
      call MPI_Comm_size(MPI_COMM_WORLD, n_proc, nErr)
      write(*, '(A, i3, A, i3)'), "myRank=", n_myRank, "n_proc=", n_proc
c-------------------------------------------------★Doutput の出力先変更
      write(strDampFile, '(A10,i3,A5)') 'DOUTPUT_S',n_myRank,'.txt'
      open (damp_out,FILE = strDampFile)
      write(damp_out, '(A10, i3)') 'rank: ', n_myRank
c------- ★★担当部材の設定
      call GetRange(1, n_myRank, n_member1) ! 担当部材の先頭番号
      call GetRange(2, n_myRank, n_member2) ! 担当部材の最終番号
       max_member=n_member2-n_member1+1     ! 担当部材の最大数
      write(damp_out,'(a,i5,a,i5,a,i5)')'max_member=',max_member,
     +                  ' n_member1=',n_member1,' n_member2=',n_member2
       ALLOCATE (pa_work_force(30,max_member))
```

上記のサブルーチンにおける出力が以下のように示される。

```
スレーブ１:      rank:   1
                 max_member=   26 n_member1=   28 n_member2=   53
スレーブ２:      rank:   2
                 max_member=   26 n_member1=   54 n_member2=   79
```

ここで示すように、2つのスレーブが立ち上がった直後に、並列環境が設定されることになる。

4.2 コントロールデータの送受信

予備計算が終了するまでに、コントロールデータや構造データ、あるいは荷重データなど多くのデータがファイルより入力される。SPACE の分散並列処理システムでは、これらのデータ入力処理をマスター側の PC が行い、その後、スレーブ全てにデータを転送する。ただし、スレーブではファイルへの入出力を実行しないので、コントロールファイルの内容であるファイル関連の情報は、スレーブ側に送信する必要はない。

最初に、解析を制御するコントロールデータの転送について説明する。コントロールデータは、次に示すマスター側のプログラムで、データが収集され、一括して送信される。データのバッファ領域として、実数データは ff_data()に、整数データは if_data()に用意されており、その個数は、iif_dt にセットされる。

```
c―――― ★★解析制御情報をファイルから入力し、スレーブに転送するためバッファにセット
c――――――――――――――――――――――――――――――――――――――――――――――――――――★解析制御情報をファイルから入力(vpp)
      ALLOCATE (ff_data(400),if_data(400),iif_dt(2))
      iif_dt(1) = 0    ! ff_data の最終場所
      iif_dt(2) = 8    ! if_data の最終場所
c――――――――――――――――――――――――――――――――――――――――――――――――――――★動的解析ダイアログその１のデータを入力(ok)
      call dyctl1_pa(ierr,NINDIT,GINDIS,F1SEC,fs_st,fl_st,ifp_st,IST,
     +               JIKUZERO,G_JIKUZERO_ALPH,
     +               ff_data(iif_dt(1)+1),if_data(iif_dt(2)+1),iif_dt)
      if(ierr.ne.0) then
      ierr_dat = 3
      call err_outf(ieer_dat)
      endif
c――――――――――――――――――――――――――――――――――――――――――――――――――――★動的解析ダイアログその２のデータを入力(ok)
      call dyctl2_pa(ierr,NSTEP,F2SEC,DELT,IGRA,IBETA,BETA,GUMMA,XGAL,
     *               NNTIME,EPSDSP,load_memb_mass,dt_M_filter,IT_ANALYS,
     +               ff_data(iif_dt(1)+1),if_data(iif_dt(2)+1),iif_dt)
      if(ierr.ne.0) then
      ierr_dat = 4
      call err_outf(ieer_dat)
      endif
c――――――――――――――――――――――――――――――――――――――――――――――――――――★解析結果の出力パラメータを入力(ok)
      call doutcl_pa(ierr,IWSTP,SOUTSC,DMAXCK,No_section,
     +                ff_data(iif_dt(1)+1),if_data(iif_dt(2)+1),iif_dt)
      if(ierr.ne.0) then
      ierr_dat = 5
      call err_outf(ieer_dat)
      endif
c――――――――――――――――――――――――――――――――――――――――――――――――――――★減衰ダイアログのデータを入力(ok)
      call damctl_pa(ierr,NREAD,ITYDP,NDMP,NDMP2,NHH,HH,QHH,
```

```
     +              ff_data(iif_dt(1)+1),if_data(iif_dt(2)+1),iif_dt)
      if(ierr.ne.0) then
      ierr_dat = 6
      call err_outf(ieer_dat)
      endif
```

上記の各サブルーチンの内容は後節で示すが、そこでは、データのパックが行われた後、以下のサブルーチン send_ctlset()で、各スレーブにコントロールデータが転送される。

```
c————————————————————————————————★制御情報をスレーブとモニターに転送(vpp)
c———— ★★　制御情報と構造用制御情報を送信
      if(n_proc.ge.2) then
      call send_ctlset(Parameter_C,ff_data,if_data,iif_dt, N_analysis)
      endif
      DEALLOCATE (ff_data,if_data,iif_dt)
```

このサブルーチンの内容についても、後節に示す。特に、送信する情報の中で、最初の８つのデータは、以下のように構造データの制御情報をセットし、送信される。このデータは、スレーブ側で構造データを受け取るための動的領域確保に利用される。

```
      id(1)=N_analysis
      id(2)=Parameter_C.n_point          !node
      id(3)=Parameter_C.n_element        !nelem
      id(4)=Parameter_C.n_member         !memb
      id(5)=Parameter_C.n_boundary_p     !nrbound
      id(6)=Parameter_C.n_local_coord    !locod
      id(7)=Parameter_C.n_rot_axis       !njiku
      id(8)=Parameter_C.n_free           !6
```

スレーブによる受信側も、同様に以下のサブルーチンでコントロールデータを受信し、受信したバッファ領域の値を、ダイアログデータを入力するサブルーチンを変更して、一旦変数にセットする。その後、これらのデータを構造体にセットし直すことになる。

```
    ALLOCATE (ff_data(400),if_data(400),iif_dt(2,4))
c———— ★★解析制御情報をマスターから取得(vpp)
      call recv_ctlset(Parameter_C,ff_data,if_data,iif_dt,N_analysis,
     *                 MPP_Ana_Group,ierr_dat)
      write(*, *) "out recv_ctlset : Parameter_C.n_element =",
     *       Parameter_C.n_element
c————————————————————————★制御情報の取得に失敗した場合はここで戻る
      if(ierr_dat .ne. 0 ) then
      DEALLOCATE (ff_data,if_data,iif_dt)
      goto 9997
      endif
```

```
c------------------------------------------------★動的解析ダイアログその１のデータをセット(ok)
      is=iif_dt(1,1)                        !  ff_data() 先頭番地
       js=iif_dt(2,1)                       !  if_data() 先頭番地
       call dyctl1_set(ierr,NINDIT,GINDIS,F1SEC,fs_st,fl_st,ifp_st,IST,
     *              JIKUZERO,G_JIKUZERO_ALPH,ff_data(is),if_data(js))
c------------------------------------------------★動的解析ダイアログその２のデータをセット(ok)
      is=iif_dt(1,2)
       js=iif_dt(2,2)
       call dyctl2_set(ierr,NSTEP,F2SEC,DELT,IGRA,IBETA,BETA,GUMMA,XGAL,
     *            NNTIME,EPSDSP,load_memb_mass,dt_M_filter,IT_ANALYS,
     +               ff_data(is),if_data(js))
c------------------------------------------------★解析結果の出力パラメータをセット(ok)
      is=iif_dt(1,3)
       js=iif_dt(2,3)
       call doutcl_set(ierr,IWSTP,SOUTSC,DMAXCK,No_section,
     +               ff_data(is),if_data(js))
c------------------------------------------------★減衰ダイアログのデータをセット(ok)
      is=iif_dt(1,4)
       js=iif_dt(2,4)
       call damctl_set(ierr,NREAD,ITYDP,NDMP,NDMP2,NHH,HH,QHH,
     +               ff_data(is),if_data(js))

      DEALLOCATE (ff_data,if_data,iif_dt)
```

以下に、例として用いたワーレン型アーチの送信用コントロールデータを示す。

```
制御データ    68    90
        8      41       1      79      41       0       0       6      32       0
        2       1       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0
        0       7       0       0       0       2      10       0       1      12
        4       0       0       0       0       0       0       0       0       0
        4       0       5       0       3       2       1       2
    63.50    0.00    0.10    0.01    0.10    0.00    0.00    0.00    0.00    0.00
     0.00    0.00    0.00  700.00  700.00    0.00    0.00    0.00    0.00    0.00
     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
     0.00    0.00    0.00    0.00   11.50    1.00    0.00  100.00  100.00  600.00
     1.00    0.00    1.00    0.20    0.50   50.00    2.50    0.00  500.00   10.50
     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
```

送られた情報を元に、構造体にデータをセットするわけであるが、ここでは、間違いなく情報が伝わっていることを示すために、このコントロールデータから設定する構造体データを、マスター側とスレーブ側の両者について以下に示す。

マスター側構造体

```
構造体：Control:
          8        =  IANAL
          0        =  in_disp
          0        =  in_stres
          0        =  NINDIT
 0.000000000000000E+000        =  GINDIS
  500.000000000000             =  DMAXCK
 0.000000000000000E+000        =  SOUTSC
          4        =  IWSTP
          1        =  IT_ANALYS
          0        =  JIKUZERO
 0.000000000000000E+000        =  G_JIKUZERO_ALPH
 Set_model_type Ok

構造体：Newmark_P
 0.100000001490116       =  t1
  1.00000000000000       =  t2
  1.10000000149012       =  t1 + t2
        100 =  t1/dt + 1
       1100 =  (t1 + t2)/dt + 1
          3 =  n_damp_type
          1 =  n_damp_1
          2 =  n_damp_2
 0.000000000000000E+000 =  bet1_1
 0.000000000000000E+000 =  bet1_2
 0.000000000000000E+000 =  bet2_1
 0.000000000000000E+000 =  bet2_2
 1.000000047497451E-003 =  dt
 0.250000000000000       =  beta
 0.500000000000000       =  delta
 9.999999747378752E-006 =  eps_v
         10 =  iroop
  1.00000000000000       =  gumma
 1.000000094994905E-006 =  dt*dt
 5.000000237487257E-004 =  delta*dt
 2.500000237487262E-007 =  beta*dt*dt
 5.000000237487257E-004 =  (1. - delta)*dt
 2.500000237487262E-007 =  (0.5- beta )*dt*dt
 5.000000474974524E-007 =  0.5*dt*dt
 Set_newmark Ok

構造体：Dynamic_load
          1  =  load_s(i)
          0  =  load_d(i)
  100.000000000000       =  amp_load_d(i)
 0.000000000000000E+000  =  dt_load_dynamic
 0.000000000000000E+000  =  dt_load_point
          0  =  load_s(i)
          0  =  load_d(i)
  100.000000000000       =  amp_load_d(i)
 0.000000000000000E+000  =  dt_load_dynamic
 0.000000000000000E+000  =  dt_load_point
          0  =  load_s(i)
```

```
           0  =  load_d(i)
  600.000000000000       =  amp_load_d(i)
0.000000000000000E+000  =  dt_load_dynamic
0.000000000000000E+000  =  dt_load_point
         0  =  load_mass
         0  =  n_load_point
         0  =  n_load_dynamic
```

スレーブ側構造体

```
構造体：Control:
          8        =  IANAL
          0        =  in_disp
          0        =  in_stres
          0        =  NINDIT
0.000000000000000E+000        =  GINDIS
 500.000000000000             =  DMAXCK
0.000000000000000E+000        =  SOUTSC
          4        =  IWSTP
          1        =  IT_ANALYS
          0        =  JIKUZERO
0.000000000000000E+000        =  G_JIKUZERO_ALPH


構造体：Newmark_P
 0.100000001490116      =  t1
  1.00000000000000      =  t2
  1.10000000149012      =  t1 + t2
       100 =  t1/dt + 1
      1100 =  (t1 + t2)/dt + 1
         3 =  n_damp_type
         1 =  n_damp_1
         2 =  n_damp_2
 0.000000000000000E+000 =  bet1_1
 0.000000000000000E+000 =  bet1_2
 0.000000000000000E+000 =  bet2_1
 0.000000000000000E+000 =  bet2_2
 1.000000047497451E-003 =  dt
 0.250000000000000      =  beta
 0.500000000000000      =  delta
 9.999999747378752E-006 =  eps_v
        10 =  iroop
  1.00000000000000      =  gumma
 1.000000094994905E-006 =  dt*dt
 5.000000237487257E-004 =  delta*dt
 2.500000237487262E-007 =  beta*dt*dt
 5.000000237487257E-004 =  (1. - delta)*dt
 2.500000237487262E-007 =  (0.5- beta )*dt*dt
 5.000000474974524E-007 =  0.5*dt*dt


構造体：Dynamic_load
          1  =  load_s(i)
          0  =  load_d(i)
```

```
   100.000000000000       =  amp_load_d(i)
0.000000000000000E+000  =  dt_load_dynamic
0.000000000000000E+000  =  dt_load_point
         0  =  load_s(i)
         0  =  load_d(i)
 100.000000000000       =  amp_load_d(i)
0.000000000000000E+000  =  dt_load_dynamic
0.000000000000000E+000  =  dt_load_point
         0  =  load_s(i)
         0  =  load_d(i)
 600.000000000000       =  amp_load_d(i)
0.000000000000000E+000  =  dt_load_dynamic
0.000000000000000E+000  =  dt_load_point
         0  =  load_mass
         0  =  n_load_point
         0  =  n_load_dynamic
```

4.3 解析データの送受信

4.3.1 構造データ

本節では、構造データの送受信について説明する。構造データは、マスターが一度データ入力した後、そのデータを整数データと実数データに分類し、パックした後、サブルーチン Send_structure()を用いて全スレーブに転送する。その際、実数データは配列 buf()に、整数データは ibuf()に保存する。その配列の大きさは、以下に示す式を用いて決定する。例えば、座標 3 とは、節点数に対する 3 倍の数を表す。データのパックは、サブルーチン Get_structure_pa()で行っている。いづれのサブルーチンの内容も次節で示すことにする。

```
c―――――― ★★　構造データの送信用バッファ領域を確保
c      バッファデータ仕様
c      buf()      １：座標 3
c                 ２：局所座標 3
c                 ３：要素 17
c                 ４：部材 4
c      ibuf()     １：境界拘束条件 7
c                 ２：要素 6
c                 ３：部材  14
c
       id_buf=Parameter_C.n_point*6+Parameter_C.n_element*17+
     *         Parameter_C.n_member*4
       jd_buf=Parameter_C.n_point*7++Parameter_C.n_element*6+
     *         Parameter_C.n_member*14
       write(damp_out, '(3i5)') Parameter_C.n_point,
     + Parameter_C.n_element, Parameter_C.n_member
       ALLOCATE (buf(id_buf+10),ibuf(jd_buf+10))
c―――――― ★★　構造データの送信用バッファ領域を確保
       call Get_structure_pa(Point,Member,Element,Parameter_C,
```

構造データの保存領域であるバッファの大きさを計算する

```
     *                    Model_type,ierr,buf,ibuf,id_buf,jd_buf)
c―――――― ★★　スレーブ、モニターに構造データを送信
      call Send_structure(buf,ibuf,id_buf,jd_buf)
      DEALLOCATE (buf,ibuf)
```

上記のサブルーチンで転送したデータは、例として用いたワーレントラス型アーチでは、以下のようである。

```
         0         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0         0
         0        -1        -1        -1        -1        -1        -1         0         0         0
        -1        -1        -1         0         0         0        -1        -1        -1         0
         0         0        -1        -1        -1         0         0         0        -1        -1
        -1         0         0         0        -1        -1        -1         0         0         0
        -1        -1        -1         0         0         0        -1        -1        -1         0
         0         0        -1        -1        -1         0         0         0        -1        -1
        -1         0         0         0        -1        -1        -1         0         0         0
        -1        -1        -1         0         0         0        -1        -1        -1         0
         0         0        -1        -1        -1         0         0         0        -1        -1
        -1         0         0         0        -1        -1        -1         0         0         0
        -1        -1        -1         0         0         0        -1        -1        -1         0
         0         0        -1        -1        -1         0         0         0        -1        -1
        -1         0         0         0        -1        -1        -1         0         0         0
        -1        -1        -1         0         0         0        -1        -1        -1         0
         0         0        -1        -1        -1         0         0         0        -1        -1
        -1         0         0         0        -1        -1        -1         0         0         0
        -1        -1        -1         0         0         0        -1        -1        -1         0
         0         0        -1        -1        -1         0         0         0        -1        -1
        -1         0         0         0        -1        -1        -1         0         0         0
        -1        -1        -1         0         0         0        -1        -1        -1         0
         0         0        -1        -1        -1         0         0         0        -1        -1
        -1         0         0         0        -1        -1        -1         0         0         0
        -1        -1        -1         0         0         0        -1        -1        -1         0
         0         0        -1        -1        -1         0         0         0        -1        -1
        -1        -1        -1        -1        -1        -1        -1         3         0         1
         0         0         0         3         1         1         2         1         1         1
         1         0         0         1         0         0         3         3         1         1
         3         1         1         1         1         0         0         1         0         0
         3         3         1         2         3         1         1         1         1         0
         0         1         0         0         3         3         1         2         4         1
         1         1         1         0         0         1         0         0         3         3
         1         3         4         1         1         1         1         0         0         1
         0         0         3         3         1         3         5         1         1         1
         1         0         0         1         0         0         3         3         1         4
         5         1         1         1         1         0         0         1         0         0
         3         3         1         4         6         1         1         1         1         0
         0         1         0         0         3         3         1         5         6         1
         1         1         1         0         0         1         0         0         3         3
         1         5         7         1         1         1         1         0         0         1
         0         0         3         3         1         6         7         1         1         1
         1         0         0         1         0         0         3         3         1         6
```

```
 8  1  1  1  1  0  0  1  0  0
 3  3  1  7  8  1  1  1  1  0
 0  1  0  0  3  3  1  7  9  1
 1  1  1  0  0  1  0  0  3  3
 1  8  9  1  1  1  1  0  0  1
 0  0  3  3  1  8 10  1  1  1
 1  0  0  1  0  0  3  3  1  9
10  1  1  1  1  0  0  1  0  0
 3  3  1  9 11  1  1  1  1  0
 0  1  0  0  3  3  1 10 11  1
 1  1  1  0  0  1  0  0  3  3
 1 10 12  1  1  1  1  0  0  1
 0  0  3  3  1 11 12  1  1  1
 1  0  0  1  0  0  3  3  1 11
13  1  1  1  1  0  0  1  0  0
 3  3  1 12 13  1  1  1  1  0
 0  1  0  0  3  3  1 12 14  1
 1  1  1  0  0  1  0  0  3  3
 1 13 14  1  1  1  1  0  0  1
 0  0  3  3  1 13 15  1  1  1
 1  0  0  1  0  0  3  3  1 14
15  1  1  1  1  0  0  1  0  0
 3  3  1 14 16  1  1  1  1  0
 0  1  0  0  3  3  1 15 16  1
 1  1  1  0  0  1  0  0  3  3
 1 15 17  1  1  1  1  0  0  1
 0  0  3  3  1 16 17  1  1  1
 1  0  0  1  0  0  3  3  1 16
18  1  1  1  1  0  0  1  0  0
 3  3  1 17 18  1  1  1  1  0
 0  1  0  0  3  3  1 17 19  1
 1  1  1  0  0  1  0  0  3  3
 1 18 19  1  1  1  1  0  0  1
 0  0  3  3  1 18 20  1  1  1
 1  0  0  1  0  0  3  3  1 19
20  1  1  1  1  0  0  1  0  0
 3  3  1 19 21  1  1  1  1  0
 0  1  0  0  3  3  1 20 21  1
 1  1  1  0  0  1  0  0  3  3
 1 20 22  1  1  1  1  0  0  1
 0  0  3  3  1 21 22  1  1  1
 1  0  0  1  0  0  3  3  1 21
23  1  1  1  1  0  0  1  0  0
 3  3  1 22 23  1  1  1  1  0
 0  1  0  0  3  3  1 22 24  1
 1  1  1  0  0  1  0  0  3  3
 1 23 24  1  1  1  1  0  0  1
 0  0  3  3  1 23 25  1  1  1
 1  0  0  1  0  0  3  3  1 24
25  1  1  1  1  0  0  1  0  0
 3  3  1 24 26  1  1  1  1  0
 0  1  0  0  3  3  1 25 26  1
 1  1  1  0  0  1  0  0  3  3
 1 25 27  1  1  1  1  0  0  1
```

```
  0   0   3   3   1  26  27   1   1   1
  1   0   0   1   0   0   3   3   1  26
 28   1   1   1   1   0   0   1   0   0
  3   3   1  27  28   1   1   1   1   0
  0   1   0   0   3   3   1  27  29   1
  1   1   1   0   0   1   0   0   3   3
  1  28  29   1   1   1   1   0   0   1
  0   0   3   3   1  28  30   1   1   1
  1   0   0   1   0   0   3   3   1  29
 30   1   1   1   1   0   0   1   0   0
  3   3   1  29  31   1   1   1   1   0
  0   1   0   0   3   3   1  30  31   1
  1   1   1   0   0   1   0   0   3   3
  1  30  32   1   1   1   1   0   0   1
  0   0   3   3   1  31  32   1   1   1
  1   0   0   1   0   0   3   3   1  31
 33   1   1   1   1   0   0   1   0   0
  3   3   1  32  33   1   1   1   1   0
  0   1   0   0   3   3   1  32  34   1
  1   1   1   0   0   1   0   0   3   3
  1  33  34   1   1   1   1   0   0   1
  0   0   3   3   1  33  35   1   1   1
  1   0   0   1   0   0   3   3   1  34
 35   1   1   1   1   0   0   1   0   0
  3   3   1  34  36   1   1   1   1   0
  0   1   0   0   3   3   1  35  36   1
  1   1   1   0   0   1   0   0   3   3
  1  35  37   1   1   1   1   0   0   1
  0   0   3   3   1  36  37   1   1   1
  1   0   0   1   0   0   3   3   1  36
 38   1   1   1   1   0   0   1   0   0
  3   3   1  37  38   1   1   1   1   0
  0   1   0   0   3   3   1  37  39   1
  1   1   1   0   0   1   0   0   3   3
  1  38  39   1   1   1   1   0   0   1
  0   0   3   3   1  38  40   1   1   1
  1   0   0   1   0   0   3   3   1  39
 40   1   1   1   1   0   0   1   0   0
  3   3   1  39  41   1   1   1   1   0
  0   1   0   0   3   3   1  40  41   1
  1   1   1   0   0   1   0   0   3
  0.00    0.00    0.00   50.00    0.00  112.39  100.00    0.00   50.43  150.00
  0.00  160.51  200.00    0.00   96.18  250.00    0.00  203.85  300.00    0.00
137.06  350.00    0.00  242.21  400.00    0.00  172.87  450.00    0.00  275.42
500.00    0.00  203.45  550.00    0.00  303.34  600.00    0.00  228.66  650.00
  0.00  325.82  700.00    0.00  248.39  750.00    0.00  342.77  800.00    0.00
262.55  850.00    0.00  354.12  900.00    0.00  271.07  950.00    0.00  359.81
1000.00   0.00  273.91 1050.00    0.00  359.81 1100.00    0.00  271.07 1150.00
  0.00  354.12 1200.00    0.00  262.55 1250.00    0.00  342.77 1300.00    0.00
248.39 1350.00    0.00  325.82 1400.00    0.00  228.66 1450.00    0.00  303.34
1500.00   0.00  203.45 1550.00    0.00  275.42 1600.00    0.00  172.87 1650.00
  0.00  242.21 1700.00    0.00  137.06 1750.00    0.00  203.85 1800.00    0.00
 96.18 1850.00    0.00  160.51 1900.00    0.00   50.43 1950.00    0.00  112.39
2000.00   0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
```

```
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00   2100.00     21.00     50.00    140.00
      2.40      0.00      0.00      0.00      0.00    120.00    120.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
```

マスター側のサブルーチンから転送されたデータは、次に示すスレーブ側のサブルーチンによって情報が伝達される。データを受け取るバッファは、先にコントロールデータの中で送られた構造制御用データによって確保される。

```
c————————————————————————————————————————★基本構造データをマスターから取得(vpp)
c———— ★★マスターから構造データを取得
c      バッファデータ仕様
c      buf()      １：座標 3
c                 ２：局所座標 3
c                 ３：要素 17
c                 ４：部材 4
c     ibuf()      １：境界拘束条件 7
c                 ２：要素 6
c                 ３：部材  14
c
       id_buf=Parameter_C.n_point*6+Parameter_C.n_element*17+
      *       Parameter_C.n_member*4
       jd_buf=Parameter_C.n_point*7++Parameter_C.n_element*6+
      *       Parameter_C.n_member*14                ! 全部材数用意する
       ALLOCATE (buf(id_buf+10),ibuf(jd_buf+10))
       write(*,*) ' allocate ',id_buf, jd_buf
c———— ★★構造データ受信
       call recv_structure(buf,ibuf,id_buf,jd_buf)
       write(*,*) ' recev_structure ok'
c———— ★★構造データをセット
       call set_structure(Point,Member,Element,Parameter_C,
      *       Model_type,ierr,buf,ibuf,n_member1,n_member2)
       write(*,*) ' set structure ok',max_member
       DEALLOCATE (buf,ibuf)
```

構造データをパックして送られた情報は、recv_structure()サブルーチンで受信し、set_structure()で元の状態に戻される。このパックしたデータは、マスター側と同じであるが、構造データにセットする部分では、このスレーブが担当する部材に関して各種の情報を作成する。

最初に、スレーブ１で受信した構造データを確認のため、以下に示す。

```
   0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0
   0   -1   -1   -1   -1   -1   -1    0    0    0
  -1   -1   -1    0    0    0   -1   -1   -1    0
   0    0   -1   -1   -1    0    0    0   -1   -1
  -1    0    0    0   -1   -1   -1    0    0    0
  -1   -1   -1    0    0    0   -1   -1   -1    0
   0    0   -1   -1   -1    0    0    0   -1   -1
  -1    0    0    0   -1   -1   -1    0    0    0
  -1   -1   -1    0    0    0   -1   -1   -1    0
   0    0   -1   -1   -1    0    0    0   -1   -1
  -1    0    0    0   -1   -1   -1    0    0    0
  -1   -1   -1    0    0    0   -1   -1   -1    0
   0    0   -1   -1   -1    0    0    0   -1   -1
  -1    0    0    0   -1   -1   -1    0    0    0
  -1   -1   -1    0    0    0   -1   -1   -1    0
   0    0   -1   -1   -1    0    0    0   -1   -1
```

```
    -1     0     0     0    -1    -1    -1     0     0     0
    -1    -1    -1     0     0     0    -1    -1    -1     0
     0     0    -1    -1    -1     0     0     0    -1    -1
    -1     0     0     0    -1    -1    -1     0     0     0
    -1    -1    -1     0     0     0    -1    -1    -1     0
     0     0    -1    -1    -1     0     0     0    -1    -1
    -1     0     0     0    -1    -1    -1     0     0     0
    -1    -1    -1     0     0     0    -1    -1    -1     0
     0     0    -1    -1    -1     0     0     0    -1    -1
    -1    -1    -1    -1    -1    -1    -1     3     0     1
     0    29   533     3     1     1     2     1     1     1
     1     0     0     1     0     0     3     3     1     1
     3     1     1     1     1     0     0     1     0     0
     3     3     1     2     3     1     1     1     1     0
     0     1     0     0     3     3     1     2     4     1
     1     1     1     0     0     1     0     0     3     3
     1     3     4     1     1     1     1     0     0     1
     0     0     3     3     1     3     5     1     1     1
     1     0     0     1     0     0     3     3     1     4
     5     1     1     1     1     0     0     1     0     0
     3     3     1     4     6     1     1     1     1     0
     0     1     0     0     3     3     1     5     6     1
     1     1     1     0     0     1     0     0     3     3
     1     5     7     1     1     1     1     0     0     1
     0     0     3     3     1     6     7     1     1     1
     1     0     0     1     0     0     3     3     1     6
     8     1     1     1     1     0     0     1     0     0
     3     3     1     7     8     1     1     1     1     0
     0     1     0     0     3     3     1     7     9     1
     1     1     1     0     0     1     0     0     3     3
     1     8     9     1     1     1     1     0     0     1
     0     0     3     3     1     8    10     1     1     1
     1     0     0     1     0     0     3     3     1     9
    10     1     1     1     1     0     0     1     0     0
     3     3     1     9    11     1     1     1     1     0
     0     1     0     0     3     3     1    10    11     1
     1     1     1     0     0     1     0     0     3     3
     1    10    12     1     1     1     1     0     0     1
     0     0     3     3     1    11    12     1     1     1
     1     0     0     1     0     0     3     3     1    11
    13     1     1     1     1     0     0     1     0     0
     3     3     1    12    13     1     1     1     1     0
     0     1     0     0     3     3     1    12    14     1
     1     1     1     0     0     1     0     0     3     3
     1    13    14     1     1     1     1     0     0     1
     0     0     3     3     1    13    15     1     1     1
     1     0     0     1     0     0     3     3     1    14
    15     1     1     1     1     0     0     1     0     0
     3     3     1    14    16     1     1     1     1     0
     0     1     0     0     3     3     1    15    16     1
     1     1     1     0     0     1     0     0     3     3
     1    15    17     1     1     1     1     0     0     1
     0     0     3     3     1    16    17     1     1     1
     1     0     0     1     0     0     3     3     1    16
```

18	1	1	1	1	0	0	1	0	0
3	3	1	17	18	1	1	1	1	0
0	1	0	0	3	3	1	17	19	1
1	1	1	0	0	1	0	0	3	3
1	18	19	1	1	1	1	0	0	1
0	0	3	3	1	18	20	1	1	1
1	0	0	1	0	0	3	3	1	19
20	1	1	1	1	0	0	1	0	0
3	3	1	19	21	1	1	1	1	0
0	1	0	0	3	3	1	20	21	1
1	1	1	0	0	1	0	0	3	3
1	20	22	1	1	1	1	0	0	1
0	0	3	3	1	21	22	1	1	1
1	0	0	1	0	0	3	3	1	21
23	1	1	1	1	0	0	1	0	0
3	3	1	22	23	1	1	1	1	0
0	1	0	0	3	3	1	22	24	1
1	1	1	0	0	1	0	0	3	3
1	23	24	1	1	1	1	0	0	1
0	0	3	3	1	23	25	1	1	1
1	0	0	1	0	0	3	3	1	24
25	1	1	1	1	0	0	1	0	0
3	3	1	24	26	1	1	1	1	0
0	1	0	0	3	3	1	25	26	1
1	1	1	0	0	1	0	0	3	3
1	25	27	1	1	1	1	0	0	1
0	0	3	3	1	26	27	1	1	1
1	0	0	1	0	0	3	3	1	26
28	1	1	1	1	0	0	1	0	0
3	3	1	27	28	1	1	1	1	0
0	1	0	0	3	3	1	27	29	1
1	1	1	0	0	1	0	0	3	3
1	28	29	1	1	1	1	0	0	1
0	0	3	3	1	28	30	1	1	1
1	0	0	1	0	0	3	3	1	29
30	1	1	1	1	0	0	1	0	0
3	3	1	29	31	1	1	1	1	0
0	1	0	0	3	3	1	30	31	1
1	1	1	0	0	1	0	0	3	3
1	30	32	1	1	1	1	0	0	1
0	0	3	3	1	31	32	1	1	1
1	0	0	1	0	0	3	3	1	31
33	1	1	1	1	0	0	1	0	0
3	3	1	32	33	1	1	1	1	0
0	1	0	0	3	3	1	32	34	1
1	1	1	0	0	1	0	0	3	3
1	33	34	1	1	1	1	0	0	1
0	0	3	3	1	33	35	1	1	1
1	0	0	1	0	0	3	3	1	34
35	1	1	1	1	0	0	1	0	0
3	3	1	34	36	1	1	1	1	0
0	1	0	0	3	3	1	35	36	1
1	1	1	0	0	1	0	0	3	3
1	35	37	1	1	1	1	0	0	1

```
       0       0       3       3       1      36      37       1       1       1
       1       0       0       1       0       0       3       3       1      36
      38       1       1       1       1       0       0       1       0       0
       3       3       1      37      38       1       1       1       1       0
       0       1       0       0       3       3       1      37      39       1
       1       1       1       0       0       1       0       0       3       3
       1      38      39       1       1       1       1       0       0       1
       0       0       3       3       1      38      40       1       1       1
       1       0       0       1       0       0       3       3       1      39
      40       1       1       1       1       0       0       1       0       0
       3       3       1      39      41       1       1       1       1       0
       0       1       0       0       3       3       1      40      41       1
       1       1       1       0       0       1       0       0       3
      0.00      0.00      0.00     50.00      0.00    112.39    100.00      0.00     50.43    150.00
      0.00    160.51    200.00      0.00     96.18    250.00      0.00    203.85    300.00      0.00
    137.06    350.00      0.00    242.21    400.00      0.00    172.87    450.00      0.00    275.42
    500.00      0.00    203.45    550.00      0.00    303.34    600.00      0.00    228.66    650.00
      0.00    325.82    700.00      0.00    248.39    750.00      0.00    342.77    800.00      0.00
    262.55    850.00      0.00    354.12    900.00      0.00    271.07    950.00      0.00    359.81
   1000.00      0.00    273.91   1050.00      0.00    359.81   1100.00      0.00    271.07   1150.00
      0.00    354.12   1200.00      0.00    262.55   1250.00      0.00    342.77   1300.00      0.00
    248.39   1350.00      0.00    325.82   1400.00      0.00    228.66   1450.00      0.00    303.34
   1500.00      0.00    203.45   1550.00      0.00    275.42   1600.00      0.00    172.87   1650.00
      0.00    242.21   1700.00      0.00    137.06   1750.00      0.00    203.85   1800.00      0.00
     96.18   1850.00      0.00    160.51   1900.00      0.00     50.43   1950.00      0.00    112.39
   2000.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00   2100.00     21.00     50.00    140.00
      2.40      0.00      0.00      0.00      0.00    120.00    120.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
```

```
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
```

上記のパックされた構造データから、スレーブ側では、以下のような構造データが作成される。ただし、スレーブ1では、担当部材が 28-53 の 26 部材である。この中で、節点データは、第1行目の第1項は節点番号、次の3項は3次元座標、次の3項は、局所座標を表す。次の行は、その節点の 6 方向の拘束状況を示す。後は、入力データ通りであるが、部材に関しては、担当部材以外は読み捨てている。

```
    節点数 =        41
    要素数 =         1
    部材数 =        79
 拘束節点数=        41
 局所座標数=         0
 回転部材数=         0
point   41
  1       0.000       0.000       0.000       0.000       0.000       0.000
 -1  -1  -1  -1  -1  -1
  2      50.000       0.000     112.386       0.000       0.000       0.000
  0   0   0  -1  -1  -1
  3     100.000       0.000      50.427       0.000       0.000       0.000
  0   0   0  -1  -1  -1
  4     150.000       0.000     160.505       0.000       0.000       0.000
  0   0   0  -1  -1  -1
  5     200.000       0.000      96.183       0.000       0.000       0.000
  0   0   0  -1  -1  -1
  6     250.000       0.000     203.845       0.000       0.000       0.000
  0   0   0  -1  -1  -1
  7     300.000       0.000     137.059       0.000       0.000       0.000
  0   0   0  -1  -1  -1
  8     350.000       0.000     242.210       0.000       0.000       0.000
  0   0   0  -1  -1  -1
  9     400.000       0.000     172.868       0.000       0.000       0.000
  0   0   0  -1  -1  -1
 10     450.000       0.000     275.424       0.000       0.000       0.000
```

節点拘束データ

節点データ

```
 0   0   0  -1  -1  -1
11     500.000       0.000     203.449       0.000       0.000       0.000
 0   0   0  -1  -1  -1
12     550.000       0.000     303.337       0.000       0.000       0.000
 0   0   0  -1  -1  -1
13     600.000       0.000     228.662       0.000       0.000       0.000
 0   0   0  -1  -1  -1
14     650.000       0.000     325.821       0.000       0.000       0.000
 0   0   0  -1  -1  -1
15     700.000       0.000     248.392       0.000       0.000       0.000
 0   0   0  -1  -1  -1
16     750.000       0.000     342.774       0.000       0.000       0.000
 0   0   0  -1  -1  -1
17     800.000       0.000     262.549       0.000       0.000       0.000
 0   0   0  -1  -1  -1
18     850.000       0.000     354.120       0.000       0.000       0.000
 0   0   0  -1  -1  -1
19     900.000       0.000     271.070       0.000       0.000       0.000
 0   0   0  -1  -1  -1
20     950.000       0.000     359.805       0.000       0.000       0.000
 0   0   0  -1  -1  -1
21    1000.000       0.000     273.914       0.000       0.000       0.000
 0   0   0  -1  -1  -1
22    1050.000       0.000     359.805       0.000       0.000       0.000
 0   0   0  -1  -1  -1
23    1100.000       0.000     271.070       0.000       0.000       0.000
 0   0   0  -1  -1  -1
24    1150.000       0.000     354.120       0.000       0.000       0.000
 0   0   0  -1  -1  -1
25    1200.000       0.000     262.549       0.000       0.000       0.000
 0   0   0  -1  -1  -1
26    1250.000       0.000     342.775       0.000       0.000       0.000
 0   0   0  -1  -1  -1
27    1300.000       0.000     248.392       0.000       0.000       0.000
 0   0   0  -1  -1  -1
28    1350.000       0.000     325.821       0.000       0.000       0.000
 0   0   0  -1  -1  -1
29    1400.000       0.000     228.662       0.000       0.000       0.000
 0   0   0  -1  -1  -1
30    1450.000       0.000     303.337       0.000       0.000       0.000
 0   0   0  -1  -1  -1
31    1500.000       0.000     203.450       0.000       0.000       0.000
 0   0   0  -1  -1  -1
32    1550.000       0.000     275.425       0.000       0.000       0.000
 0   0   0  -1  -1  -1
33    1600.000       0.000     172.869       0.000       0.000       0.000
 0   0   0  -1  -1  -1
34    1650.000       0.000     242.211       0.000       0.000       0.000
 0   0   0  -1  -1  -1
35    1700.000       0.000     137.059       0.000       0.000       0.000
 0   0   0  -1  -1  -1
36    1750.000       0.000     203.846       0.000       0.000       0.000
 0   0   0  -1  -1  -1
37    1800.000       0.000      96.184       0.000       0.000       0.000
```

```
  0    0    0   -1   -1   -1
 38     1850.000         0.000       160.506         0.000         0.000         0.000
  0    0    0   -1   -1   -1
 39     1900.000         0.000        50.428         0.000         0.000         0.000
  0    0    0   -1   -1   -1
 40     1950.000         0.000       112.387         0.000         0.000         0.000
  0    0    0   -1   -1   -1
 41     2000.000         0.000         0.001         0.000         0.000         0.000
 -1   -1   -1   -1   -1   -1
elements     1
  1    3    2100.00      21.00      50.00
  3    1
model check     3    3    3
element:    1    3    1
membs    79   28   53
 28    1     3     1    14    16     1     3
 29    2     3     1    15    16     1     3
 30    3     3     1    15    17     1     3
 31    4     3     1    16    17     1     3
 32    5     3     1    16    18     1     3
 33    6     3     1    17    18     1     3
 34    7     3     1    17    19     1     3
 35    8     3     1    18    19     1     3
 36    9     3     1    18    20     1     3
 37   10     3     1    19    20     1     3
 38   11     3     1    19    21     1     3
 39   12     3     1    20    21     1     3
 40   13     3     1    20    22     1     3
 41   14     3     1    21    22     1     3
 42   15     3     1    21    23     1     3
 43   16     3     1    22    23     1     3
 44   17     3     1    22    24     1     3
 45   18     3     1    23    24     1     3
 46   19     3     1    23    25     1     3
 47   20     3     1    24    25     1     3
 48   21     3     1    24    26     1     3
 49   22     3     1    25    26     1     3
 50   23     3     1    25    27     1     3
 51   24     3     1    26    27     1     3
 52   25     3     1    26    28     1     3
 53   26     3     1    27    28     1     3
```

部材データ

部材データでは、左が解析対象の構造用部材番号であり、その次の番号が、このスレーブが担当する部材の番号を示す。また、その次が、部材の部材モデル番号であり、次が要素番号を示す。さらに、次の２つが当該部材に接続する節点番号を示す。節点に関連する情報は、全スレーブで、マスターと同一の情報を持つことになるので、このデータは変更する必要はない。

ファイバーデータなどの弾塑性情報を保存する動的領域として、マスター側は部材データをそのまま使用するため全部材について持つ必要

があるが、スレーブ側は担当部材分の動的領域を用意し、使用することになる。そのため、スレーブ側では、動的領域を大幅に削減することが可能となっている。

4.3.2 初期不整データ

形状初期不整は、スレーブ側の解析でも必要となるため、次に示すマスター側のコードで、初期不整データをファイルより読み込み、データをパックした後、全スレーブにデータを転送する。例題に用いた解析モデルでは、初期不整を使用していないため、各サブルーチンを示すことでその内容を説明する。処理内容は非常に単純なので理解は容易である。以下は、並列処理用マスター側の主プログラムの一部であり、初期不整データ入力とデータ転送用サブルーチンである。入力用サブルーチンは二度コールされる。最初のコールで、初期不整を有する節点数を読み込み、このサブルーチンを抜けた後、この値を用いて転送用バッファを動的確保する。

```
c―――――――――――――――――――――――――――――――――――――――――――――――――――――★初期不整データを入力(ok)
      if(Control.init_imperfection .ne. 0) then
      call Get_imperfection_pa(ihan,Control.amp_imperfection,Point,
     *     Parameter_C,buf,npoint_imp,ii1_buf)
       ihan=1
       id_buf=npoint_imp*4+1
       write(damp_out, '(a, i5, a, i5)')
     +"npoint_imp=", npoint_imp, " id_buf=", id_buf
       ALLOCATE (buf(id_buf+10))
      call Get_imperfection_pa(ihan,Control.amp_imperfection,Point,
     *     Parameter_C,buf,npoint_imp,ii1_buf)
c――――― ★★　スレーブに初期不整データを転送
      call Send_imperfection(buf,ii1_buf,npoint_imp)
      DEALLOCATE (buf)
      Enfif
```

初期不整をデータ入力するサブルーチン Get_imperfection_pa()を以下に示す。

```
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●　SUBROUTINE /Get_imperfection(チェック OK)
C       ―――――――――――――――――――――――――――――――――――――――――――――――――
C     ●　初期不整データを入力し、節点座標値に加える(ok)
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――
      subroutine Get_imperfection_pa(ihan,amp_imperfection,Point,
     *           Parameter_C,buf,npoint,ii1)
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      record / parameter_s  / Parameter_C
```

```
       record / point_s      / Point
      dimension Point(*)
      dimension buf(*)
C     ------------------------------------------------------------------
c     amp_imperfection :real*8 初期不整の大きさ
c     Parameter_C      :structure
c     Point            :structure
C     ------------------------------------------------------------------
      if(ihan.eq.0) then
      read(5,*) npoint
      else
      ii1=0
      if(npoint.eq.0) return
      do i=1,npoint
      read(5,*) i1,am1,am2,am3
      Point(i1).disp_initial(1) =  am1 * amp_imperfection
      Point(i1).disp_initial(2) =  am2 * amp_imperfection
      Point(i1).disp_initial(3) =  am3 * amp_imperfection
      ii1=ii1+1
      buf(ii1)=i1+0.5
      ii1=ii1+1
      buf(ii1)=am1* amp_imperfection
      ii1=ii1+1
      buf(ii1)=am2* amp_imperfection
      ii1=ii1+1
      buf(ii1)=am3* amp_imperfection
      end do
      do i=1,Parameter_C.n_point
      do j=1,3
      Point(i).coord(j) = Point(i).coord(j)+Point(i).disp_initial(j)
      end do
      end do
      endif
      return
      end
```

さらに、転送用サブルーチン Send_imperfection()を示す。

```
C     ------------------------------------------------------------------
C     ●   SUBROUTINE /Send_imperfection
C       ----------------------------------------------------------------
C     ●   初期不整データを入力し、節点座標値に加える(ok)
C     ------------------------------------------------------------------
      subroutine Send_imperfection(buf,ii1,npoint)
c 外部宣言
      use MPI_DEFINE
c 引数宣言
      real*8 :: buf(*)
      integer ii1, npoint
c 実装
c     初期不整データをスレーブに転送
c------- ★★  制御情報と構造用制御情報を送信
      mpi_buf = loc(npoint)          ! 変数をバッファに格納
```

```
      call mpi_bcast(npoint,1,MPI_INTEGER
     *              ,ID_MASTER,MPI_COMM_WORLD,ierr)
      call mpi_bcast(buf,ii1,MPI_DOUBLE
     *              ,ID_MASTER,MPI_COMM_WORLD,ierr)
      return
      end
```

データは２つに分けて転送する。最初は、初期不整を有する節点数であり、後は、実際の不整データである。ここでは、ii1 は転送データ数、npoint は初期不整を有する節点数である。

次に、スレーブ側のデータ受信に関するコードを示す。受信用データ数はこの時点では分からないので、解析モデルの節点数を用いて確保する。

```
c―――――――――――――――――――――――――――――――――――――――――――――――――――――★初期不整データをマスターから取得(ok)
c――――― ★★マスターから初期不整データを取得
      if(Control.init_imperfection .ne. 0) then
      n_inperfection = 4*Parameter_C.n_point +1      !    初期不整の節点が不明のため節点使用
      ALLOCATE (buf(n_inperfection+10))
c     write(*,*) ' recv_imperfection in',n_inperfection
      call recv_imperfection(Point,buf, n_inperfection)
c     write(*,*) ' recv_imperfection out',Element(1).element_type
      DEALLOCATE (buf)
```

初期不整データの受信用サブルーチン recv_imperfection()を以下に示す。この中で、受信した初期不整データ（初期変位）を解析モデルの節点座標に加えている。

```
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●    SUBROUTINE /recv_imperfection
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●    初期不整データを入力し、節点座標値に加える(ok)
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――
      subroutine recv_imperfection(Point,buf,iix)
c 外部宣言
      use MPI_DEFINE
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
C     引数宣言
      record / point_s      / Point
      dimension Point(*)
      real*8:: buf(*)
      integer:: iix
c 実装
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――
c
c           マスターより初期不整データを取得
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――
```

```
       write(*, *) "recv_imperfection"
c———— ★★  制御情報と構造用制御情報を送信        ★★————
      call MPI_Bcast(npoint,1, MPI_INTEGER,
     *                        ID_MASTER, MPI_COMM_WORLD, ierr)
      iix=npoint*4
      call MPI_Bcast(buf, iix, MPI_DOUBLE,
     *                        ID_MASTER, MPI_COMM_WORLD, ierr)
c———— ★★  ———————————————————————— ★★————
      ii1=0
      write(76,'(a,i4)') ' imperfection ',npoint
      do i1=1,npoint
      ii1=ii1+1
      i=buf(ii1)
      do j=1,3
      ii1=ii1+1
      Point(i).coord(j) = Point(i).coord(j)+buf(ii1)
      write(76,'(3i4,3f10.3)') i1,i,j,buf(ii1)
      end do
      enddo
      return
      end
```

4.3.3 特殊断面（ファイバー）データ

本節では、特殊断面（ファイバー）データの入力と送受信について説明する。次に示すコードで、ファイバーデータをファイルより読み込み、データをパックした後、全スレーブにデータを転送する。データ入力を行い、送信用にデータをバッファにパックするサブルーチンFiber_input_pa()は二度コールされる。一度目のコールで、バッファ用データの大きさ id_buf と jd_buf を決定する。このサブルーチンについては、次節で説明する。

```
      call Fiber_input_pa(0,ierr,Parameter_C.n_member,
     *      Parameter_C.n_element,Member,Element,Model_type,
     *      E_model_fiber,M_model_fiber,E_model11,M_model11,
     *      E_model12,M_model12,E_model13,M_model13,
     *      E_model15,M_model15,
     *      E_model21,M_model21,E_model22,M_model22,
     *      E_model31,M_model31,E_model32,M_model32,
     *      E_model33,M_model33,
     *      id_buf,jd_buf,buf,ibuf )
c———— ★★スレーブにファイバー用バッファをゼロセット
      ALLOCATE (buf(id_buf+10),ibuf(jd_buf+10))
      do i=1,id_buf+10
      buf(i)=0.
      enddo
      do i=1,jd_buf+10
      ibuf(i)=0
      enddo
      call Fiber_input_pa(1,ierr,Parameter_C.n_member,
```

```
      *       Parameter_C.n_element,Member,Element,Model_type,
      *       E_model_fiber,M_model_fiber,E_model11,M_model11,
      *       E_model12,M_model12,E_model13,M_model13,
      *       E_model15,M_model15,
      *       E_model21,M_model21,E_model22,M_model22,
      *       E_model31,M_model31,E_model32,M_model32,
      *       E_model33,M_model33,
      *       id_buf,jd_buf ,buf,ibuf)

c―――― ★★スレーブにファイバーデータを転送
      call Send_Fiber(id_buf,jd_buf ,buf,ibuf)
       DEALLOCATE (buf,ibuf)
```

スレーブに転送するサブルーチン Send_Fiber()を以下に示す。ここでは、３つに分けてデータを全スレーブに転送する。内容は簡単なので理解は容易である。

```
C     ―――――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Send_Fiber
C       ―――――――――――――――――――――――――――――――――――――――
C     ●   ファイバー要素のデータをスレーブに転送(ok)
C     ―――――――――――――――――――――――――――――――――――――――

C     ―――――――――――――――――――――――――――――――――――――――
      subroutine Send_Fiber(id_buf,jd_buf ,buf,ibuf)
c 外部宣言
      use MPI_DEFINE
      implicit real*8(A-H,O-Z)
c 引数宣言
      real*8 :: buf(*)
      integer:: ibuf(*)
      integer :: id_buf, jd_buf
c 内部変数宣言
      integer :: iibuf(2)
c 実装
! ファイバーデータの個数を送る
      iibuf(1)=id_buf
      iibuf(2)=jd_buf
      call mpi_bcast(iibuf,2, MPI_INTEGER,
     +               ID_MASTER,MPI_COMM_WORLD,ierr)
! ファイバーデータ(整数)を送る
      call mpi_bcast(ibuf,jd_buf, MPI_INTEGER,
     +               ID_MASTER,MPI_COMM_WORLD,ierr)
      write(76,'(a,i4)') ' fiber ',jd_buf
      write(76,'(10i4)')(ibuf(i),i=1,jd_buf)
! ファイバーデータ(実数)を送る
      call mpi_bcast(buf,id_buf, MPI_DOUBLE,
     +               ID_MASTER,MPI_COMM_WORLD,ierr)
      write(76,'(a,i4)') ' fiber ',id_buf
      write(76,'(10f10.2)')(buf(i),i=1,id_buf)
      return
       end
```

スレーブ側には、一回目で整数と実数データの個数が送られており、その値を用いて実際のデータを受け取るバッファの大きさを決める。次に、サブルーチン recv_Fiber()が二度コールされ、一度目のコールでファイバーデータが受信される。

```
c———— ★★マスターからファイバー用制御データを取得
      call recv_Fiber_P(id_buf,jd_buf)
c———— ★★スレーブにファイバー用バッファをゼロセット
       ALLOCATE (buf(id_buf+10),ibuf(jd_buf+10))
      call recv_Fiber(0,ierr,Parameter_C.n_member,
     *      Parameter_C.n_element,Member,Element,Model_type,
     *      E_model_fiber,M_model_fiber,E_model11,M_model11,E_model12,
     *      M_model12,E_model13,M_model13,E_model15,M_model15,
     *      E_model21,M_model21,E_model22,M_model22,
     *      E_model31,M_model31,E_model32,M_model32,E_model33,M_model33,
     *       id_buf,jd_buf,buf,ibuf)
c————————————————————★ファイバーモデルデータをマスターから取得
c———— ★★マスターからファイバー用データを取得
      call recv_Fiber(1,ierr,Parameter_C.n_member,
     *      Parameter_C.n_element,Member,Element,Model_type,
     *      E_model_fiber,M_model_fiber,E_model11,M_model11,E_model12,
     *      M_model12,E_model13,M_model13,E_model15,M_model15,
     *      E_model21,M_model21,E_model22,M_model22,
     *      E_model31,M_model31,E_model32,M_model32,E_model33,M_model33,
     *       id_buf,jd_buf,buf,ibuf)
```

以下に示すサブルーチン recv_Fiber_P()では、ファイバーデータをパックしたデータの整数データと実数データの個数を受信している。

```
C     ——————————————————————————————————————————
C     ●    SUBROUTINE /recv_Fiber_P
C       ————————————————————————————————————————
C     ●     ファイバー用バッファ領域の大きさを取得
C     ——————————————————————————————————————————
      subroutine recv_Fiber_P(id_buf,jd_buf)
c 外部宣言
      use MPI_DEFINE
c 引数宣言
integer:: id_buf                  ! 実数バッファの大きさ
integer:: jd_buf                  ! 整数バッファの大きさ
c 内部変数
integer if_data(2)                ! 受信バッファ
c 実装
c———— ★★  制御情報と構造用制御情報を受信      ★★————
      call MPI_Bcast(if_data, 2, MPI_INTEGER,
     +                ID_MASTER, MPI_COMM_WORLD, ierr)
      id_buf=if_data(1)
      jd_buf=if_data(2)
      return
       end
```

サブルーチン recv_Fiber()は二度コールされるが、一度目に実際のファイバーデータが受信される。受信バッファの大きさは、上記のサブルーチンで受信した値を用いる。以下に、受信部分のコードを示す。他の部分の内容については次節で示す。

```
C     ------------------------------------------------------------------
C     ●   SUBROUTINE /recv_Fiber
C       ----------------------------------------------------------------
C     ●   ファイバー要素の入力(ok)
C     ------------------------------------------------------------------
      subroutine recv_Fiber(it,ierr,n_member,n_element,Member,
     *      Element,Model_type,E_model_fiber,M_model_fiber,
     *      E_model11,M_model11,E_model12,M_model12,E_model13,M_model13,
     *      E_model15,M_model15,
     *      E_model21,M_model21,E_model22,M_model22,
     *      E_model31,M_model31,E_model32,M_model32,E_model33,M_model33,
     *      id_buf,jd_buf,buf,ibuf)
c 外部宣言
      use MPI_DEFINE
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      include "..¥..¥sf3st¥submainx.h"
c 引数宣言
      integer:: it        ! 制御コード(0:データ受信,  1:データセット)
      integer :: ierr, n_member, n_element
      record / member_s          / Member
      record / element_s         / Element
      record / E_model11_s       / E_model11
      record / E_model12_s       / E_model12
      record / E_model13_s       / E_model13
      record / E_model15_s       / E_model15
      record / E_model21_s       / E_model21
      record / E_model22_s       / E_model22
      record / E_model31_s       / E_model31
      record / E_model32_s       / E_model32
      record / E_model33_s       / E_model33
      record / M_model11_s       / M_model11
      record / M_model12_s       / M_model12
      record / M_model13_s       / M_model13
      record / M_model15_s       / M_model15
      record / M_model21_s       / M_model21
      record / M_model22_s       / M_model22
      record / M_model31_s       / M_model31
      record / M_model32_s       / M_model32
      record / M_model33_s       / M_model33
      record / E_model_fiber_s   / E_model_fiber
      record / M_model_fiber_s   / M_model_fiber
      record / n_model_s          / Model_type
      dimension E_model_fiber(*),M_model_fiber(*)
      dimension E_model11(*),M_model11(*),E_model12(*),M_model12(*),
     *          E_model15(*),M_model15(*),E_model13(*),M_model13(*)
      dimension E_model21(*),M_model21(*),E_model22(*),M_model22(*)
```

```
      dimension E_model31(*),M_model31(*),E_model32(*),M_model32(*)
      dimension E_model33(*),M_model33(*)
      dimension Member(*),Element(*)
      dimension ddm(20)
      real*8:: buf(*)
      integer :: ibuf(*)
C     ------------------------------------------------------------------
C     実装
      ierr=0
      if(it.eq.0) then
      write(76,*) ' MPI_Bcast in 1',jd_buf
      write(76,*) 'n_member= ', n_member, 'n_element= ', n_element
c------- ★★  制御情報と構造用制御情報を受信        ★★-------
      call MPI_Bcast(ibuf, jd_buf, MPI_INTEGER,
     +               ID_MASTER, MPI_COMM_WORLD, ierr)
      write(76,'(10i4)') ( ibuf(i),i=1,jd_buf)
      write(76, '(A, i5)') 'id_buf =', id_buf
      call MPI_Bcast(buf, id_buf, MPI_DOUBLE,
     +              ID_MASTER, MPI_COMM_WORLD, ierr)
      write(76,'(10f10.2)') ( buf(i),i=1,id_buf)
      write(76,*) ' mpi_Bcast out'
c-------------------------------------------------★ファイバーデータの予備入力
      ii=0
      jd_buf=1                                  ! 並列用
      id_buf=0                                  ! 並列用
      nm = ibuf(1)
      write(76,'(a,i4)') ' fiber number:',nm
      do i=1,nm
      jd_buf=jd_buf+2                           ! 並列用
      n_m=ibuf(jd_buf-1)                        ! 並列用
      nmm=ibuf(jd_buf)                          ! 並列用
       .
       .
```

4.3.4 RO モデルデータ

本節では、RO モデルデータの送受信について説明する。ここでも、データ入力用サブルーチン RO_data_input_pa()は二度コールされる。一度目で、送信用バッファの大きさを求め、サブルーチンを抜けた後、バッファ領域を動的に確保する。さらに、二度目で入力したデータをバッファ領域にパックする。

```
      call RO_data_input_pa(0,n,RO_work,Element,Parameter_C.n_element,
     *      ierr,id_buf,jd_buf,buf,ibuf )
       ALLOCATE (buf(id_buf+10))
       call RO_data_input_pa(1,n,RO_work,Element,Parameter_C.n_element,
     *      ierr,id_buf,jd_buf,buf,ibuf )
c------- ★★スレーブに RO モデル用データを転送
      call Send_RO_data(id_buf,jd_buf,buf)
      DEALLOCATE (buf)
```

具体的に、サブルーチン RO_data_input_pa()の内容を、以下に示す。

```
C     ------------------------------------------------------------------
C     ●    SUBROUTINE /RO_data_input
C       ----------------------------------------------------------------
C     ●    RO モデルデータ入力(ok)
C     ------------------------------------------------------------------
      subroutine RO_data_input_pa(it,n,RO_work,Element,n_element,ierr,
     *      id_buf,jd_buf,buf,ibuf )
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      include "..¥..¥sf3st¥submainx.h"
      record / RO_work_s     /  RO_work
      record / element_s2_RO / Element
      dimension RO_work(*),Element(*)
       integer RO_MODEL_NUMBER,TRI_MODEL_NUMBER
       real*8 BI_DEF
       integer      n_ro
       dimension buf(*),ibuf(*)
       data    BI_MODEL_NUMBER/11/
      data   RO_MODEL_NUMBER/12/
c
       BI_DEF=10.0**(-10.0)
c-------------------------------------------------------★データ入力(ok)
       if(it.eq.0) then
       ierr=0
       read(5,*,end=999,err=998) nn
       id_buf=nn*27
        jd_buf=nn
       if(nn.le.0) goto 998
        else
        nn=jd_buf
        id_buf=0
        do i=1,nn
              read(5,*,end=999,err=998) (RO_work(i).gan(j),  j=1,3),
     *                                    (RO_work(i).arf1(j),j=1,6),
     *                                    (RO_work(i).arf2(j),j=1,6),
     *                                    (RO_work(i).beta(j),j=1,6),
     *                                    (RO_work(i).anyu(j),j=1,6)
         do j=1,3
        id_buf=id_buf+1
        buf(id_buf)=RO_work(i).gan(j)
        enddo
         do j=1,6
        id_buf=id_buf+1
        buf(id_buf)=RO_work(i).arf1(j)
        enddo
         do j=1,6
        id_buf=id_buf+1
        buf(id_buf)=RO_work(i).arf2(j)
        enddo
         do j=1,6
```

```
        id_buf=id_buf+1
        buf(id_buf)=RO_work(i).beta(j)
        enddo
         do j=1,6
        id_buf=id_buf+1
        buf(id_buf)=RO_work(i).anyu(j)
        enddo
       enddo
c------------------------------------------------------------★部材の線形剛性計算(ok)
c     Element(i).AKu          ! 軸剛性
c     Element(i).AK_1         ! 線形せん断剛性
c
       ii=0
       do i=1,n_element
              if(Element(i).element_type.eq.2) then
                     if(    Element(i).nm_type.eq.RO_MODEL_NUMBER.or.
     *              Element(i).nm_type.eq.BI_MODEL_NUMBER)then
                            ii=ii+1
                            if(ii.gt.nn) goto 999

c-------------------------★初期剛性の計算挿入箇所(後で積層ゴム厚をかける必要があり)
                            n_ro=Element(i).No
                            if(Element(i).nm_type.eq.BI_MODEL_NUMBER)then
                                   Element(i).AK_1
     *                             =0.7*RO_work(n_ro).arf1(1)*BI_DEF**(-0.3)
                            else if(Element(i).nm_type.eq.RO_MODEL_NUMBER)then
                                   Element(i).AK_1
     *                             =Element(i).Ar*RO_work(n_ro).arf1(1)
                            endif

                     endif
              endif
       enddo
       endif

       return

  998 continue
       ierr=1
       return

  999 continue
       ierr=2
       return
       end
```

次に、データを送信するサブルーチン Send_RO_data()を以下に示す。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /Send_RO_data
C       ----------------------------------------------------------
C     ●   RO モデルデータをスレーブに転送(ok)
C     ------------------------------------------------------------
      subroutine Send_RO_data(id_buf,jd_buf,buf)
```

```
c 外部宣言
      use MPI_DEFINE
      implicit real*8(A-H,O-Z)
c 引数宣言
      real*8:: buf(*)
      integer :: id_buf, jd_buf
c 実装
      call mpi_bcast(jd_buf,1,MPI_INTEGER,
     +               ID_MASTER,MPI_COMM_WORLD,ierr)
      call mpi_bcast(buf,id_buf, MPI_DOUBLE,
     +               ID_MASTER,MPI_COMM_WORLD,ierr)
      return
      end
```

スレーブ側で、データの受信を以下のサブルーチンで行う。このサブルーチンでも二度のコールが行われ、一度目のサブルーチンコールで受信用バッファの大きさを取得する。

```
c―――― ★★マスターから修正 RO モデル用データを取得
      call recv_RO_data(0,n,RO_work,Element,Parameter_C.n_element,
     *                          ierr,id_buf,jd_buf,buf,ibuf)
c      if(ierr .ne. 0 ) goto 9997
      ALLOCATE (buf(id_buf+10))
      call recv_RO_data(1,n,RO_work,Element,Parameter_C.n_element,
     *                          ierr,id_buf,jd_buf,buf,ibuf)
      DEALLOCATE (buf)
```

受信サブルーチンの内容を以下に示す。

```
C     ―――――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /recv_RO_data
C       ―――――――――――――――――――――――――――――――――――――――
C     ●   RO モデルデータ入力(ok)
C     ―――――――――――――――――――――――――――――――――――――――
      subroutine recv_RO_data(it,n,RO_work,Element,n_element,ierr,
     *                                      id_buf,jd_buf,buf,ibuf)
c 外部宣言
      use MPI_DEFINE
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      include "..¥..¥sf3st¥submainx.h"
C     引数宣言
      integer:: it
      record / RO_work_s     /  RO_work
      record / element_s2_RO / Element
      dimension RO_work(*), Element(*)
      integer:: n_element, ierr, id_buf, jd_buf
      real*8:: buf(*)
      integer:: ibuf(*)
c 内部変数
```

```
        integer RO_MODEL_NUMBER,TRI_MODEL_NUMBER
       real*8  BI_DEF
       integer n_ro
      data  BI_MODEL_NUMBER/11/
       data  RO_MODEL_NUMBER/12/
c 実装
       BI_DEF=10.0**(-10.0)
c-------------------------------------------------------★データ入力(ok)
       if(it.eq.0) then
       ierr=0
c       read(5,*,end=999,err=998) nn
       call MPI_Bcast(jd_buf,1, MPI_INTEGER,
     +                ID_MASTER, MPI_COMM_WORLD, ierr)
       nn=jd_buf
c-------------------------------------------------------★個数データ受信(ok)
       id_buf=nn*27
c      jd_buf=nn
       if(nn.le.0) goto 998
       else
       call MPI_Bcast(buf, id_buf, MPI_DOUBLE,
     +              ID_MASTER, MPI_COMM_WORLD, ierr)
       id_buf=0
       nn=jd_buf
       do i=1,nn
c     read(5,*,end=999,err=998) (RO_work(i).gan(j), j=1,3),
c      *(RO_work(i).arf1(j),j=1,6),
c      *(RO_work(i).arf2(j),j=1,6),
c      *(RO_work(i).beta(j),j=1,6),
c      *(RO_work(i).anyu(j),j=1,6)
       do j=1,3
       id_buf=id_buf+1
       RO_work(i).gan(j)=buf(id_buf)
       enddo
       do j=1,6
       id_buf=id_buf+1
       RO_work(i).arf1(j)=buf(id_buf)
       enddo
       do j=1,6
       id_buf=id_buf+1
       RO_work(i).arf2(j)=buf(id_buf)
       enddo
       do j=1,6
       id_buf=id_buf+1
       RO_work(i).beta(j)=buf(id_buf)
       enddo
       do j=1,6
       id_buf=id_buf+1
       RO_work(i).anyu(j)=buf(id_buf)
       enddo
       enddo
c-------------------------------------------------------★部材の線形剛性計算(ok)
c     Element(i).AKu          ! 軸剛性
c     Element(i).AK_1         ! 線形せん断剛性
c
```

```
       ii=0
       do i=1,n_element
       if(Element(i).element_type.eq.2) then
      if(Element(i).nm_type.eq.RO_MODEL_NUMBER.or.
     *  Element(i).nm_type.eq.BI_MODEL_NUMBER)then
       ii=ii+1
       if(ii.gt.nn) goto 999
c------------------------★初期剛性の計算挿入箇所(後で積層ゴム厚をかける必要があり)
       n_ro=Element(i).No
       if(Element(i).nm_type.eq.BI_MODEL_NUMBER)then
       Element(i).AK_1
     *    =0.7*RO_work(n_ro).arf1(1)*BI_DEF**(-0.3)
       else if(Element(i).nm_type.eq.RO_MODEL_NUMBER)then
       Element(i).AK_1
     *    =Element(i).Ar*RO_work(n_ro).arf1(1)
       endif
       endif
       endif
       enddo
       endif
       return

  998 continue
       ierr=1
       return

  999 continue
       ierr=2
       return
       end
```

4.3.5 レーリー減衰データ

本節では、レーリー減衰で使用するパラメータの送受信について説明する。送信すべきパラメータは、既に求められており、構造体にセットされている。このパラメータを次のサブルーチンによって全スレーブに転送する。送信データ個数は一定であるので、この送信用サブルーチンコールは一回である。

```
c------- ★★スレーブに減衰用データを転送
       call Send_damp(Newmark_P,ierr)
```

減衰用データの送信用サブルーチンの内容は以下の用である。

```
C     ------------------------------------------------------------
C     ●    SUBROUTINE /Send_damp
C       ----------------------------------------------------------
C     ●     減衰データをセットする(ok)
C     ------------------------------------------------------------
      subroutine Send_damp(Newmark_P,ierr)
```

```
c 外部宣言
      use MPI_DEFINE
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      parameter(damp_out = 76)                ! 計算結果のダンプ出力ファイル番号
c 引数の宣言
      record / newmark_s  / Newmark_P
c 内部変数の宣言
      real*8 :: buf(4)
c 実装
      buf(1)=Newmark_P.alf1_1
      buf(2)=Newmark_P.alf1_2
      buf(3)=Newmark_P.alf2_1
      buf(4)=Newmark_P.alf2_2
      write(76,'(a,4f12.4)') '  damp ',(buf(i),i=1,4)
C     ------------------------------------------------------------------
c                減衰定数をスレーブに転送
C     ------------------------------------------------------------------
      call MPI_Bcast(buf,4, MPI_DOUBLE,
     +               ID_MASTER,MPI_COMM_WORLD,ierr)
      return
       end
```

上記のサブルーチンによる減衰データは、以下のようである。送信データは最後の４つのデータである。

```
        レーリー減衰             3
         Newmark_P.alf1_1:    1.00000000000000
         Newmark_P.alf1_2:    1.00000000000000
         Newmark_P.alf2_1:   2.999999932944775E-002
         Newmark_P.alf2_2:   2.999999932944775E-002
         Newmark_P.n_damp_1:             1
         Newmark_P.n_damp_2:             2
         OMG1_1:    36.8857010000000
         OMG1_2:    101.674300000000
         OMG2_1:    36.8857010000000
         OMG2_2:    101.674300000000
         Newmark_P.alf1_1:    54.1329070744493
         Newmark_P.alf1_2:   1.443418003439535E-002
         Newmark_P.alf2_1:    1.62398717593454
         Newmark_P.alf2_2:   4.330253913529886E-004
          Get_damp Ok
          damp       54.1329       0.0144       1.6240       0.0004
```

スレーブ側の受信用サブルーチンコールは、以下のようである。

```
c------------------------------------------------★レーリー減衰をマスターから取得(ok)
c--------- ★★マスターから減衰データを取得
c      write(*,*) ' recv_damp in'
       call recv_damp(Newmark_P,ierr)
```

このスレーブ側の受信用サブルーチンの内容を示す。

```
C     ------------------------------------------------------------------
C     ●    SUBROUTINE /recv_damp
C     ------------------------------------------------------------------
C     ●    初期不整データを入力し、節点座標値に加える(ok)
C     ------------------------------------------------------------------
      subroutine recv_damp(Newmark_P,ierr)
c 外部変数
      use MPI_DEFINE
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
C     引数宣言
      record / newmark_s  / Newmark_P
C     内部変数
      real*8 buf(6)
c 実装
C     ------------------------------------------------------------------
c
c          マスターより減衰データを取得
C     ------------------------------------------------------------------
      do i=1,4
      buf(i)=0
      enddo
      call MPI_Bcast(buf,4, MPI_DOUBLE,
     +               ID_MASTER, MPI_COMM_WORLD, ierr)
      Newmark_P.alf1_1=buf(1)
      Newmark_P.alf1_2=buf(2)
      Newmark_P.alf2_1=buf(3)
      Newmark_P.alf2_2=buf(4)
      write(76,'(a,4f12.4,i4)') ' damp ',(buf(i),i=1,4)
      return
      end
```

このサブルーチンによって、次のようにデータが受け渡される。このデータを用いて、スレーブ側のレーリー減衰用パラメータが設定される。

```
 damp       54.1329       0.0144       1.6240       0.0004
```

以上で、解析用入力データが全て設定され、マスターから全スレーブに対し、これらの解析データが送信されたことになる。

4.4 解析時のデータ転送

動的ソルバーの主要部分であるニューマークβ法において、マスターとスレーブ間でデータの送受信が必要になってくる。このデータの送受信が最小限のデータで、しかも最も効率良く行うことが、分散並列型動

4.4.1 マスター側の加速度データ転送のためのテーブル作成

的解析の計算効率向上に寄与する。

動的解析時のデータ送受信は、以下に示す通りであるが、データの大きさはスレーブの担当部材によって決定され、各スレーブごと異なっている。スレーブでは、必要最小限の加速度データをマスターから受信し、また最小限の右辺項データと部材の応力ベクトルを送信する。マスター側では、解析ステップごとに、各スレーブに加速度ベクトルを各スレーブで必要最小限のデータとして、テーブルを利用してパックし、送信する。逆に、スレーブから送られてくる右辺項ベクトルは、パックされているため、これを同じテーブルを利用して元の状態に戻しながら、右辺項ベクトルに足しこむことになる。これらの操作法については、次節で示す。

以下に示す送受信のデータで、ＭとＳは、各々マスター側とスレーブ側のシステムを表す。また、記号－＞はデータの転送方向を表す。

１．　加速度データ（Ｍ　－＞　Ｓ）
２．　右辺項ベクトル（Ｓ　－＞　Ｍ）
３．　部材の応力ベクトル（Ｓ　－＞　Ｍ）
４．　制御用データ（Ｍ　－＞　Ｓ）

Ver.1.11 では、４のスレーブ制御用データは、１の加速度データに含まれる

上記のデータ転送の説明に入る前に、各スレーブに合わせて、データをパックする方法について解説する。まず、各スレーブに転送するデータをパックするためのテーブルを作成する。このテーブル Ms_table は加速度のスレーブ転送用であり、次のコードで作成される。また、各スレーブにおけるテーブル Ms_table の最大自由度は、配列 Ms_free にセットされる。

```
c―――――― ★★加速度のスレーブ転送用テーブル作成
      if(n_proc.ge.0) then
      write(damp_out,*) ' Convert_node_inf_vpp in Ok',n_unknown
      call Convert_node_inf_vpp(Parameter_C.n_unknown,Parameter_C,
     *      Member,Point,Ms_table,Ms_free,n_proc,nm_parallel)
      write(damp_out,*) ' Convert_node_inf_vpp out Ok'
      do i=1, n_proc-1
      write(damp_out,'(a,i3,a,i10)') 'ms_free(',i,')=',Ms_free(i)
      end do
      endif
```

上記サブルーチンの内容を以下に示す。ここで、n_proc は分散並列型システムで実際に稼動しているプロセスの数である。

```
C―――――――――――――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Convert_node_inf_vpp(マスター用)
```

```
C     ―――――――――――――――――――――――――――――――――
C       ●    マスターとスレーブ間の変換用テーブル作成
C―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
        subroutine Convert_node_inf_vpp(n_unknown,Parameter_C,Member,
     *       Point,Ms_table,Ms_free,n_proc,nm_parallel )
        implicit real*8(a-h,o-z)
        include "..¥..¥sf3st¥submain.h"
        record /member_s        / Member
        record /parameter_s     / Parameter_C
        record /point_s/ Point
        dimension Member(*),Point(*)
        dimension Ms_table(n_unknown,*),Ms_free(*)
        integer nm_parallel(2,0:n_proc)
        integer, ALLOCATABLE ::table_W(:)

        ALLOCATE (table_W(Parameter_C.n_point))
        write(76,'(a,3i8)')' n_proc ',n_proc,Parameter_C.n_point
        do k = 1, n_proc-1
        n_member1 = nm_parallel(1,k  )                !  担当部材の初期値
        n_member2 = nm_parallel(2,k  )                !  担当部材の最終値
        max_member=n_member2-n_member1+1              !  担当部材の最大数
        write(76,'(a,3i4)')' proc ',k,n_member1,n_member2
        call Create_table(Parameter_C,Member,Point,Ms_free(k),
     *       Ms_table(1,k),table_W,n_member1,n_member2)
        enddo
        write(76,'(a,i8)')' n_proc end',n_proc
        DEALLOCATE (table_W)
        write(76,'(a,i8)')' deallocate',n_proc
        return
        end
C―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
C       ●    SUBROUTINE /Create_table(マスター用)
C     ―――――――――――――――――――――――――――――――――
C       ●    マスター用圧縮テーブル作成
C―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
        subroutine Create_table(Parameter_C,Member,Point,is_free,
     *       Ms_table,table_W,n_member1,n_member2)
        implicit real*8(a-h,o-z)
        include "..¥..¥sf3st¥submain.h"
        record /member_s        / Member
        record /parameter_s     / Parameter_C
        record /point_s         / Point
        dimension Member(*),Point(*)
        dimension Ms_table(*)
        integer table_W(*)
c                                       ワークテーブルをゼロクリアする
        do i=1,Parameter_C.n_point
        table_W(i)=0
        enddo
c                                       担当部材の両端の節点番号のワークテーブルに１をセット
        do i=n_member1,n_member2
        do j=1,2
        is = Member(i).nm_point(j)
        table_W(is)=1
```

```
      enddo
      enddo
c                                          ワークテーブルが１のとき、自由度番号をセット
      is_free=0
      do i=1,Parameter_C.n_point
      if(table_W(i).ne.0) then
      do j=1,6
      if(Point(i).irest(j).gt.0) then
      is_free=is_free+1
      Ms_table(is_free)=Point(i).irest(j)
      write(76,'(a,5i4)') 'ms_table ',i,j,is_free,Ms_table(is_free)
      endif
      enddo
      endif
      enddo
      return
      end
```

上記のサブルーチンによって作られた加速度転送用テーブルを以下に示す。このテーブルは、図 4-1 に示すワーレントラス型アーチについて作成したものである。この例ではプロセスは 3 つで、マスターと 2 つのスレーブで PC クラスターが構成される。スレーブ 1 では、担当部材は、28 から 53 である。このテーブルに含まれる節点は担当部材に連結している節点の全てであり、また、このテーブルの値は、転送する加速度のマスター側の未知番号を示す。

このテーブルでは、例えば Ms_table 14 は 14 節点で、次の 1 はその節点の自由度番号（x 方向変位)、次の 1 は、テーブルの番号を示す。最後の 25 は、テーブルの値であり、これがマスター側の未知番号を示す。

```
n_proc        3      41
proc    1  28  53
ms_table   14   1   1  25
ms_table   14   3   2  26
ms_table   15   1   3  27
ms_table   15   3   4  28
ms_table   16   1   5  29
ms_table   16   3   6  30
ms_table   17   1   7  31
ms_table   17   3   8  32
ms_table   18   1   9  33
ms_table   18   3  10  34
ms_table   19   1  11  35
ms_table   19   3  12  36
ms_table   20   1  13  37
ms_table   20   3  14  38
ms_table   21   1  15  39
ms_table   21   3  16  40
```

```
ms_table   22   1  17  41
ms_table   22   3  18  42
ms_table   23   1  19  43
ms_table   23   3  20  44
ms_table   24   1  21  45
ms_table   24   3  22  46
ms_table   25   1  23  47
ms_table   25   3  24  48
ms_table   26   1  25  49
ms_table   26   3  26  50
ms_table   27   1  27  51
ms_table   27   3  28  52
ms_table   28   1  29  53
ms_table   28   3  30  54
```

次が、スレーブ2 に対するテーブルである。

```
 proc    2  54  79
ms_table   27   1   1  51
ms_table   27   3   2  52
ms_table   28   1   3  53
ms_table   28   3   4  54
ms_table   29   1   5  55
ms_table   29   3   6  56
ms_table   30   1   7  57
ms_table   30   3   8  58
ms_table   31   1   9  59
ms_table   31   3  10  60
ms_table   32   1  11  61
ms_table   32   3  12  62
ms_table   33   1  13  63
ms_table   33   3  14  64
ms_table   34   1  15  65
ms_table   34   3  16  66
ms_table   35   1  17  67
ms_table   35   3  18  68
ms_table   36   1  19  69
ms_table   36   3  20  70
ms_table   37   1  21  71
ms_table   37   3  22  72
ms_table   38   1  23  73
ms_table   38   3  24  74
ms_table   39   1  25  75
ms_table   39   3  26  76
ms_table   40   1  27  77
ms_table   40   3  28  78
```

このテーブルを利用して、スレーブへのデータ転送を行う。以下にはマスター側で管理している節点の未知番号表を示す。

```
n_unknown:          78
        1       0       0       0       0       0       0
        2       1       0       2       0       0       0
```

3	3	0	4	0	0	0
4	5	0	6	0	0	0
5	7	0	8	0	0	0
6	9	0	10	0	0	0
7	11	0	12	0	0	0
8	13	0	14	0	0	0
9	15	0	16	0	0	0
10	17	0	18	0	0	0
11	19	0	20	0	0	0
12	21	0	22	0	0	0
13	23	0	24	0	0	0
14	25	0	26	0	0	0
15	27	0	28	0	0	0
16	29	0	30	0	0	0
17	31	0	32	0	0	0
18	33	0	34	0	0	0
19	35	0	36	0	0	0
20	37	0	38	0	0	0
21	39	0	40	0	0	0
22	41	0	42	0	0	0
23	43	0	44	0	0	0
24	45	0	46	0	0	0
25	47	0	48	0	0	0
26	49	0	50	0	0	0
27	51	0	52	0	0	0
28	53	0	54	0	0	0
29	55	0	56	0	0	0
30	57	0	58	0	0	0
31	59	0	60	0	0	0
32	61	0	62	0	0	0
33	63	0	64	0	0	0
34	65	0	66	0	0	0
35	67	0	68	0	0	0
36	69	0	70	0	0	0
37	71	0	72	0	0	0
38	73	0	74	0	0	0
39	75	0	76	0	0	0
40	77	0	78	0	0	0
41	0	0	0	0	0	0

4.4.2 スレーブ側の部材未知番号表作成

本節では、スレーブ側の加速度データにアクセスするための未知番号表を作成するサブルーチンについて説明する。スレーブに転送される加速度ベクトルは、必要最小限にパックされているため、元の状態に戻して使用する必要がある。しかし、ここでは加速度ベクトルはパックしたままの状態で使用することとし、逆に、加速度ベクトルに直接アクセスするため、スレーブ用の部材未知番号表を作り直して使用する。ここでは、以下のコードで部材両端の未知番号表を作る。

```
c————————————————————————————————★部材両端の拘束表作成
c———— ★★節点拘束表の変更（スレーブ）
      call Reset_ij_table(Parameter_C,Member,Point,
     *       Parameter_C.n_unknown)
c        write(*,*) ' Set_restraint_member Ok'
```

サブルーチン Reset_ij_table()の内容を以下に示す。ここでは、既に部材数は担当部材数となっており、構造体 Member の中身は、担当部材のデータのみで設定されている。

```
C————————————————————————————————————————————————————
C     ●   SUBROUTINE /Reset_ij_table(スレーブ用)
C     ————————————————————————
C     ●   未知番号の取り替え
c         スレーブ側     部材番号と部材データは担当部材に変更
c                        節点番号と節点座標は変更せず
c                        釣合式の右辺項は担当部材に関連するものに圧縮
C————————————————————————————————————————————————————
      subroutine Reset_ij_table(Parameter_C,Member,Point,is_free)
      implicit real*8(a-h,o-z)
      include "..¥..¥sf3st¥submain.h"
      record /member_s       / Member
      record /parameter_s    / Parameter_C
      record /point_s/ Point
      integer, ALLOCATABLE ::table_W(:)
      dimension Member(*),Point(*)

      ALLOCATE (table_W(Parameter_C.n_point))
      n_member= Parameter_C.n_member   ! 既に担当部材数に変更済み
c                                ワーク用テーブルをゼロクリア
      do i=1,Parameter_C.n_point
      table_W(i)=0
      enddo
c                                部材両端に連結している節点のワークテーブルを１にセット
      do i=1,n_member
      do j=1,2
      is = Member(i).nm_point(j)
      table_W(is)=1
      enddo
      enddo
c                                節点拘束表を作成し直す
      is_free=0
      do i=1,Parameter_C.n_point
      if(table_W(i).ne.0) then
      do j=1,6
      if(Point(i).irest(j).gt.0) then
      is_free=is_free+1
      Point(i).irest(j)=is_free
      endif
      enddo
      endif
      enddo
C                                 節点拘束表から部材両端の拘束表を作成
```

```
write(76,'(a)') ' スレーブ部材両端の拘束表'
do i=1,n_member
is = Member(i).nm_point(1)
do k=1,6
Member(i).irest(k)= Point(is).irest(k)
enddo
is = Member(i).nm_point(2)
do k=1,6
Member(i).irest(k+6)= Point(is).irest(k)
enddo
write(76,'(i4,6i6,5x,6i6)') i,(Member(i).irest(j),j=1,12)
enddo
DEALLOCATE (table_W)

return
end
```

上記プログラムで作成した部材両端の未知番号表を以下に示す。この未知番号表は、スレーブ１における表であり、担当部材は、28 から 53 までの 26 部材である。

スレーブ１における部材両端の加速度データ検索テーブル（未知番号表）

1	1	0	2	0	0	0	5	0	6	0	0	0
2	3	0	4	0	0	0	5	0	6	0	0	0
3	3	0	4	0	0	0	7	0	8	0	0	0
4	5	0	6	0	0	0	7	0	8	0	0	0
5	5	0	6	0	0	0	9	0	10	0	0	0
6	7	0	8	0	0	0	9	0	10	0	0	0
7	7	0	8	0	0	0	11	0	12	0	0	0
8	9	0	10	0	0	0	11	0	12	0	0	0
9	9	0	10	0	0	0	13	0	14	0	0	0
10	11	0	12	0	0	0	13	0	14	0	0	0
11	11	0	12	0	0	0	15	0	16	0	0	0
12	13	0	14	0	0	0	15	0	16	0	0	0
13	13	0	14	0	0	0	17	0	18	0	0	0
14	15	0	16	0	0	0	17	0	18	0	0	0
15	15	0	16	0	0	0	19	0	20	0	0	0
16	17	0	18	0	0	0	19	0	20	0	0	0
17	17	0	18	0	0	0	21	0	22	0	0	0
18	19	0	20	0	0	0	21	0	22	0	0	0
19	19	0	20	0	0	0	23	0	24	0	0	0
20	21	0	22	0	0	0	23	0	24	0	0	0
21	21	0	22	0	0	0	25	0	26	0	0	0
22	23	0	24	0	0	0	25	0	26	0	0	0
23	23	0	24	0	0	0	27	0	28	0	0	0
24	25	0	26	0	0	0	27	0	28	0	0	0
25	25	0	26	0	0	0	29	0	30	0	0	0
26	27	0	28	0	0	0	29	0	30	0	0	0

以下には、マスター側で使用している全部材の未知番号表を示す。

部材未知番号表　79

1	0	0	0	0	0	0	1	0	2	0	0	0
2	0	0	0	0	0	0	3	0	4	0	0	0
3	1	0	2	0	0	0	3	0	4	0	0	0
4	1	0	2	0	0	0	5	0	6	0	0	0
5	3	0	4	0	0	0	5	0	6	0	0	0
6	3	0	4	0	0	0	7	0	8	0	0	0
7	5	0	6	0	0	0	7	0	8	0	0	0
8	5	0	6	0	0	0	9	0	10	0	0	0
9	7	0	8	0	0	0	9	0	10	0	0	0
10	7	0	8	0	0	0	11	0	12	0	0	0
11	9	0	10	0	0	0	11	0	12	0	0	0
12	9	0	10	0	0	0	13	0	14	0	0	0
13	11	0	12	0	0	0	13	0	14	0	0	0
14	11	0	12	0	0	0	15	0	16	0	0	0
15	13	0	14	0	0	0	15	0	16	0	0	0
16	13	0	14	0	0	0	17	0	18	0	0	0
17	15	0	16	0	0	0	17	0	18	0	0	0
18	15	0	16	0	0	0	19	0	20	0	0	0
19	17	0	18	0	0	0	19	0	20	0	0	0
20	17	0	18	0	0	0	21	0	22	0	0	0
21	19	0	20	0	0	0	21	0	22	0	0	0
22	19	0	20	0	0	0	23	0	24	0	0	0
23	21	0	22	0	0	0	23	0	24	0	0	0
24	21	0	22	0	0	0	25	0	26	0	0	0
25	23	0	24	0	0	0	25	0	26	0	0	0
26	23	0	24	0	0	0	27	0	28	0	0	0
27	25	0	26	0	0	0	27	0	28	0	0	0
28	25	0	26	0	0	0	29	0	30	0	0	0
29	27	0	28	0	0	0	29	0	30	0	0	0
30	27	0	28	0	0	0	31	0	32	0	0	0
31	29	0	30	0	0	0	31	0	32	0	0	0
32	29	0	30	0	0	0	33	0	34	0	0	0
33	31	0	32	0	0	0	33	0	34	0	0	0
34	31	0	32	0	0	0	35	0	36	0	0	0
35	33	0	34	0	0	0	35	0	36	0	0	0
36	33	0	34	0	0	0	37	0	38	0	0	0
37	35	0	36	0	0	0	37	0	38	0	0	0
38	35	0	36	0	0	0	39	0	40	0	0	0
39	37	0	38	0	0	0	39	0	40	0	0	0
40	37	0	38	0	0	0	41	0	42	0	0	0
41	39	0	40	0	0	0	41	0	42	0	0	0
42	39	0	40	0	0	0	43	0	44	0	0	0
43	41	0	42	0	0	0	43	0	44	0	0	0
44	41	0	42	0	0	0	45	0	46	0	0	0
45	43	0	44	0	0	0	45	0	46	0	0	0
46	43	0	44	0	0	0	47	0	48	0	0	0
47	45	0	46	0	0	0	47	0	48	0	0	0
48	45	0	46	0	0	0	49	0	50	0	0	0
49	47	0	48	0	0	0	49	0	50	0	0	0
50	47	0	48	0	0	0	51	0	52	0	0	0
51	49	0	50	0	0	0	51	0	52	0	0	0
52	49	0	50	0	0	0	53	0	54	0	0	0
53	51	0	52	0	0	0	53	0	54	0	0	0

54	51	0	52	0	0	0	55	0	56	0	0	0
55	53	0	54	0	0	0	55	0	56	0	0	0
56	53	0	54	0	0	0	57	0	58	0	0	0
57	55	0	56	0	0	0	57	0	58	0	0	0
58	55	0	56	0	0	0	59	0	60	0	0	0
59	57	0	58	0	0	0	59	0	60	0	0	0
60	57	0	58	0	0	0	61	0	62	0	0	0
61	59	0	60	0	0	0	61	0	62	0	0	0
62	59	0	60	0	0	0	63	0	64	0	0	0
63	61	0	62	0	0	0	63	0	64	0	0	0
64	61	0	62	0	0	0	65	0	66	0	0	0
65	63	0	64	0	0	0	65	0	66	0	0	0
66	63	0	64	0	0	0	67	0	68	0	0	0
67	65	0	66	0	0	0	67	0	68	0	0	0
68	65	0	66	0	0	0	69	0	70	0	0	0
69	67	0	68	0	0	0	69	0	70	0	0	0
70	67	0	68	0	0	0	71	0	72	0	0	0
71	69	0	70	0	0	0	71	0	72	0	0	0
72	69	0	70	0	0	0	73	0	74	0	0	0
73	71	0	72	0	0	0	73	0	74	0	0	0
74	71	0	72	0	0	0	75	0	76	0	0	0
75	73	0	74	0	0	0	75	0	76	0	0	0
76	73	0	74	0	0	0	77	0	78	0	0	0
77	75	0	76	0	0	0	77	0	78	0	0	0
78	75	0	76	0	0	0	0	0	0	0	0	0
79	77	0	78	0	0	0	0	0	0	0	0	0

4.4.3 右辺項データの送受信

本節では、スレーブ側で計算した動的釣合式の右辺項、ただし、担当部材に関するデータをマスター側に送信し、また、このデータをマスター側で受信するサブルーチンについて説明する。右辺項は、既に作り直した部材両端の未知番号を用いて、パックした状態の右辺項が作成されている。この右辺項をマスター側に転送するために、スレーブ側では、次のコードを用いる。

```
c————————————————————————————————★右辺項を送る（vpp）
c———— ★★マスターに右辺項を送信
      call send_ld_repeat(n_unknown,ld_point_repeat)
```

サブルーチン send_ld_repeat()の内容を以下に示す。MPI_c 関数を用いて右辺項ベクトルである ld_point_repeat を送信する。

```
C————————————————————————————————————————————
C     ●   SUBROUTINE /send_ld_repeat(スレーブ用)
C     ————————————————————
C     ●   右辺項をマスターに送る
```

```
C-----------------------------------------------------------------------
      subroutine send_ld_repeat(n_unknown,ld_point_repeat)
c 外部宣言
      use MPI_DEFINE
      implicit none
c 引数宣言
      integer:: n_unknown                ! 未知数の数
      real*8::ld_point_repeat(*)         ! マスターに送る右辺項
c 内部変数
      integer:: ierr
c 実装
      call mpi_send(ld_point_repeat, n_unknown, MPI_DOUBLE,
     +              ID_MASTER, MPI_ANY_TAG, MPI_COMM_WORLD, ierr)
      return
      end
```

マスター側では、各スレーブから転送された右辺項のパック状態を展開し、マスターの右辺項に加える処理を行う。この処理を行うコードは以下のようである。

```
c------- ★★スレーブより右辺項を受信
c      write(damp_out, *) 'n_proc=', n_proc
       if( n_proc .ge. 2)then
c------- ★★  Time Check Codes
      call MPI_WTime(TIME_NET_LD, nStart)
c-----------------------------------------------
      call recv_ld_repeat(n_unknown,ld_point_repeat,
     +                    Ms_table,Ms_free,n_proc)
     endif
```

サブルーチン recv_ld_repeat()の内容を示す。データの受信は MPI_c 関数を用いており、スレーブ数(n_proc－1)分行われる。受信データ数は、各スレーブで異なっており、Ms_free()にその値が保存されている。各スレーブでパックされている右辺項をサブルーチン Dt_from_S_to_M()を用いて展開し、その値をマスター側で計算している右辺項に足しこんでいる。その際、各スレーブ用の Ms_table を用いて行われている。

```
C-----------------------------------------------------------------------
C     ●   SUBROUTINE /recv_ld_repeat
C     ------------------------------
C     ●   各プロセスが計算した圧縮右辺項を集め、展開して足し込む
C-----------------------------------------------------------------------
      subroutine recv_ld_repeat(n_unknown,ld_point_repeat,
     +                          Ms_table,Ms_free,n_proc)
c 外部宣言
      use MPI_DEFINE
```

```
      implicit none
C 引数宣言
      integer:: n_unknown            ! 未知数
      real*8:: ld_point_repeat(*)    ! 受け取る右辺項
      integer:: Ms_table(n_unknown,*)! 変換テーブル
      integer:: Ms_free(*)
      integer:: n_proc               ! プロセス総数
C 内部変数宣言
      real*8, ALLOCATABLE :: buf_ld(:)       ! 受信バッファ
      integer:: recv_status(MPI_STATUS_SIZE) ! 受信ステータス
      integer nproc_unknown                  ! そのプロセスの担当している未知数の数
      integer:: ierr                         ! エラー番号
c     integer :: kk                          ! M-S 部材番号変換テーブルの番号(ID_SLAVE から１番とする)
      integer:: k,i
c 実装
      ALLOCATE (buf_ld(n_unknown))
      do k=1,n_proc-1
      nproc_unknown = Ms_free(k)
                                             ! 右辺項を受信
      call MPI_Recv(buf_ld, nproc_unknown, MPI_DOUBLE,
     +              k, MPI_ANY_TAG, MPI_COMM_WORLD, recv_status,ierr)
c                                    テーブルを用いて buf_ld → ld_point_repeat へ足しこむ
      call Dt_from_S_to_M(buf_ld,ld_point_repeat,
     *                    nproc_unknown,Ms_table(1,k))
      enddo
      DEALLOCATE (buf_ld)
      return
      end
C-----------------------------------------------------------------------
C     ●   SUBROUTINE /Dt_from_S_to_M(マスター用)
C     ------------------------------------
C     ●   スレーブからマスターに圧縮したデータを解除してセット
C-----------------------------------------------------------------------
      subroutine Dt_from_S_to_M(disp_S,disp_M,is_free,Ms_table)
      implicit real*8(a-h,o-z)
C-----------------------------------------------------------------------
      dimension disp_S(*),disp_M(*),Ms_table(is_free)
      integer is_free
C-----------------------------------------------------------------------
      do i=1,is_free
      j=ms_table(i)
      disp_M(j)=disp_M(j)+disp_S(i)
      enddo
      return
      end
```

4.4.4 加速度データの送受信

ここでは、マスター側の増分加速度ベクトルを、各スレーブが必要とする最小限のデータにパックし、送信する。送信用のサブルーチンコールは以下のようである。

```
c――――――――――――――――――――――――――――――――――――――――★加速度をスレーブに送信(ok)
c―――――― ★★スレーブに計算結果を転送
      if( n_proc .ge. 2)then
c―――――― ★★  Time Check Codes
         call MPI_WTime(TIME_NET_ACC, nStart)
c――――――――――――――――――――――――――――――――――――――――
c        write(damp_out, '(a, i5)') 'Send_Result_acc:istep = ', istep
         call Send_Results_acc(n_unknown,result_acc_point,
     *      nm_parallel,n_proc, Ms_table, Ms_free)
```

上記のサブルーチンの内容を示す。

```
C――――――――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Send_Results_acc(マスター用)
C     ―――――――――――――――――――――――――――――――
C     ●   マスターが計算した加速度を圧縮し、スレーブに転送
C――――――――――――――――――――――――――――――――――――――――――――――――――――――――
      subroutine Send_Results_acc(n_unknown,result_acc_point,
     *         nm_parallel, n_proc, Ms_table, Ms_free)
c 外部宣言
      use MPI_DEFINE
      implicit none
c 引数宣言
      integer:: n_unknown              ! マトリクスの未知数の数
      real*8:: result_acc_point(*)     ! 解析モデル全体の加速度ベクトル
      integer :: n_proc                ! 総プロセス数
      integer :: nm_parallel(2,0:n_proc)! 部材担当範囲
      integer :: Ms_table(n_unknown, *) ! データ変換用テーブル
      integer :: Ms_free(*)            ! 各スレーブの最大自由度
c 内部変数
      real*8, ALLOCATABLE ::disp_S(:)  ! 変換後のスレーブに送信する加速度
      integer:: k                      ! カウンタ
      integer:: ierr
c 実装
      ALLOCATE (disp_S(n_unknown))
      do k = 1, n_proc-1
      call Dt_from_M_to_S(disp_S,result_acc_point,
     *                  Ms_free(k),Ms_table(1,k))
      call MPI_Send(disp_S, Ms_free(k), MPI_DOUBLE, k,
     +                  MPI_ANY_TAG, MPI_COMM_WORLD, ierr)
      enddo
      DEALLOCATE (disp_S)
      return
      end
```

Ver.1.11 では、加速度ベクトルの最後にスレーブ制御用データを含むため、このサブルーチンは変更される

上記プログラムで理解できるように、スレーブの数(n_proc – 1)だけ、加速度をサブルーチン Dt_from_M_to_S()で配列 disp_S にパックし、サブルーチン MPI_Send()でスレーブにデータを転送する。データをパックするサブルーチン Dt_from_M_to_S()を以下に示す。

```
C-----------------------------------------------------------------------
C      ●    SUBROUTINE /Dt_from_M_to_S(マスター用)
C      ------------------------------------
C      ●    マスターからスレーブに圧縮したデータを送る
C-----------------------------------------------------------------------
       subroutine Dt_from_M_to_S(disp_S,disp_M,is_free,Ms_table)
       implicit real*8(a-h,o-z)
       dimension disp_S(*),disp_M(*),Ms_table(*)
      do i=1,is_free
       disp_S(i)=disp_M(ms_table(i))
       enddo
       return
       end
```

データのパックは、各スレーブ用の Ms_table テーブルと最大自由度 Ms_free(k)を用いることによって、非常に単純な方法で行っている。

次に、スレーブ側における加速度受信コードを以下に示す。

```
c-------------------------------------------------★サーバーで計算した加速度を受け取る（vpp）
c-------- ★★マスターから計算結果加速度を取得
            call recv_result_acc(n_unknown,result_acc_point,ierr)
```

このサブルーチンの内容を以下に示す。受信用データの個数は、スレーブ側の全自由度と同じであることから、受信用個数を先に転送する必要はない。そこで、MPI_c 関数を用いてデータを受信する。

```
C-----------------------------------------------------------------------
C      ●    SUBROUTINE /recv_result_acc(スレーブ用)
C      ------------------------------
C      ●    計算加速度をマスターから受ける
C-----------------------------------------------------------------------
       subroutine recv_result_acc(n_unknown,result_acc_point, ierr)
c 外部宣言
       use MPI_DEFINE
       implicit none
C       引数宣言
       integer:: n_unknown                    ! 未知数
       real*8::result_acc_point(*)            ! 加速度ベクトル
       integer:: ierr
C 内部変数
       integer:: recv_status(MPI_STATUS_SIZE) ! 受信ステータス
c 実装
       call MPI_Recv(result_acc_point, n_unknown, MPI_DOUBLE,
      +             ID_MASTER, MPI_ANY_TAG,MPI_COMM_WORLD, recv_status, ierr)
       return
       end
```

Ver.1.11 では、加速度ベクトルの最後にスレーブ制御用データを含むため、このサブルーチンは変更される

4.4.5 応力データの送受信

本節では、スレーブ側で計算した応力データをマスター側に転送する。

転送データ個数は各スレーブでの担当部材数で決められる。スレーブ側における応力データ転送コードは以下のようである。ここで分かるように、この応力データはマスター側で、部材応力をファイルに出力するためであり、各スレーブで計算したデータを必要とするためにデータ転送するわけである。そのため、出力要求を制御するパラメータ i_print に従ってデータ送信を行っている。

```
c――――――――――――――――――――――――――――――――――――――――――――――――――――★部材の節点力と部材応力をマスターに送る
c―――――――― ★★マスターに部材応力を送信（マスターがファイル出力する時のみ）
       if(i_print.eq.0) then
      call send_force_vpp(n_member,Member,pa_work_force)
      endif
```

応力データ転送サブルーチン send_force_vpp()の内容を以下に示す。この中で、部材の応力と部材両端の節点力、並びに部材両端と中央の弾塑性状態を表すパラメータを転送用バッファ領域にセットしている。その後、MPI_c 関数を用いてマスター側にデータを送信する。

```
C―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●    SUBROUTINE /send_force_vpp（スレーブ用）
C     ―――――――――――――――――――――――――――――――――
C     ●   各スレーブが計算した節点力をマスターに送る
C―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
      subroutine send_force_vpp(n_member,Member, pa_work_force)
c 外部宣言
      use MPI_DEFINE
      implicit none
      include ″..¥..¥sf3st¥submain.h″
c 引数宣言
      integer:: n_member
      record / member_s   / Member(*)
      real*8:: pa_work_force(33,*)
c 内部変数
      integer:: n_count                   ! 配列の数
      integer:: i,j
      integer:: ierr
c 本体
      n_count=n_member*33
      do i=1,n_member
      do j=1,12
      pa_work_force(j,i)=Member(i).force(j)
      end do
      do j=1,18
      pa_work_force(j+12,i)=Member(i).stress(j)
      end do
      do j=1,3
      pa_work_force(j+30,i)=Member(i).d_stat(j)
      enddo
```

```
        end do
        call MPI_Send(pa_work_force,n_count,MPI_DOUBLE,
       +              ID_MASTER, MPI_ANY_TAG, MPI_COMM_WORLD, ierr)
        return
        end
```

応力データをマスター側で受信するためのコードが以下に示されている。ここでは、スレーブ側に合わせて出力制御パラメータ i_print に従って受信コードが実行される。

```
c―――――― ★★部材両端の節点力を受け取る
      if(i_print.eq.0) then                          ! ファイル出力のときのみデータを取得
      if( n_proc .ge. 2)then
c―――――― ★★  Time Check Codes
      call MPI_WTime(TIME_NET_FORCE, nStart)
c―――――――――――――――――――――――――――――――――――――――
      write(damp_out, '(a, i5)') 'recv_force_vpp:istep = ', istep
      call recv_force_vpp( Parameter_C,Member,pa_work_force,
     +                                    nm_parallel,n_proc)
```

応力受信用サブルーチン recv_force_vv()の内容を以下に示す。応力データを受信した後、スレーブで行った処理と逆の処理を行い、各部材の応力を分担した部材の構造体 Member にセットする。

```
C―――――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /recv_force_vpp
C     ―――――――――――――――――
C     ●  各スレーブが計算した節点力をスレーブから受け取る
C―――――――――――――――――――――――――――――――――――――――――――――――――――――
      subroutine recv_force_vpp(Parameter_C,
     +                    Member,pa_work_force, nm_parallel, n_proc)
c 外部宣言
      use MPI_DEFINE
      implicit none
      include "..¥..¥sf3st¥submain.h"
c 引数宣言
      record /parameter_s    / Parameter_C
c      Member                 structure
c     ld_point_repeat(*)    : real*8   非線形右辺項ベクトル
c     pa_work_stress        : real*8   応力の転送用領域
c     n_member1             : integer  担当部材の先頭番号
c     n_member2             : integer  担当部材の最終番号
c     n_id                  : integer  プロセスナンバー
      record / member_s   / Member
      dimension Member(*)
      real*8:: pa_work_force(33,*)
      integer :: n_proc
      integer :: nm_parallel(2, 0:n_proc)
c 内部変数宣言
      integer::n_member1,n_member2
```

```
       integer, save :: n_Maxcount
      integer :: recv_status(MPI_STATUS_SIZE)  ! 受信ステータスを宣言
      integer:: k,i,j
      integer:: ii
      integer:: ierr
c 実装
      do k=1,n_proc-1
      n_Maxcount=Parameter_C.n_member*30
      n_member1 = nm_parallel(1,k  )
      n_member2 = nm_parallel(2,k  )
      n_Maxcount=(n_member2-n_member1+1)*33    !  担当部材の最大数
      call MPI_Recv(pa_work_force,n_Maxcount, MPI_DOUBLE,
     +                k, MPI_ANY_TAG, MPI_COMM_WORLD, recv_status,ierr)
      do i=n_member1,n_member2
      ii=i-n_member1+1
      do j=1,12
      Member(i).force(j)=pa_work_force(j,ii)
      end do
      do j=1,18
      Member(i).stress(j)=pa_work_force(j+12,ii)
      end do
      do j=1,3
      Member(i).d_stat(j)=pa_work_force(j+30,ii)+0.1
      enddo
      end do
      enddo
      return
       end
```

4.4.6 最大応力の送受信

解析が終了するとマスター側では、応力の最大値をファイルに出力する。応力の最大値は、各部材で管理しており、スレーブ側にも存在する。そこで、動的解析が終了すると、各スレーブよりマスターに担当部材の応力最大値が転送されることになる。

以下にスレーブよりマスターに応力最大値を送信するコードを示す。

```
c-----------------------------------------------------------------------
c
c     ★          応答解析終了処理
c
c-----------------------------------------------------------------------
 9998 continue
c-------- ★★  最大応力等を送信
      call send_max_stress(Parameter_C,Max_stress)
 9997 continue
```

逆に、この最大値をマスター側では、次のコードで受信する。次のサブルーチンでファイルに応力の最大値を出力することになる。

```
c     write(damp_out,*) ' Out_max_disp ok'
c———— ★★  最大応力等をスレーブからの受信
      if( n_proc .ge. 2)then
      call recv_max_stress( Parameter_C,Max_stress,
     +            nm_parallel,n_proc)
      endif
c————————————————————————★最大応力等の出力
      call Out_max_stress(Max_stress,Parameter_C.n_member,
     *                   ifl(16),iflz(16))
c     write(damp_out,*) ' Out_max_stress ok'
```

上記の応力の最大値を送信、受信するサブルーチンの内容を以下に示す。

```
C——————————————————————————————————————
C     ●   SUBROUTINE /send_max_stress
C     ——————————————————
C     ●   各プロセスが計算した最大応力を取得
C——————————————————————————————————————
      subroutine send_max_stress( Parameter_C,Max_stress)
c 外部宣言
      use MPI_DEFINE
      implicit real*8(a-h,o-z)
      include "..¥..¥sf3st¥submain.h"
c 引数宣言
      record /parameter_s    / Parameter_C
      record / max_stress_s  / Max_stress
      dimension Max_stress(*)
c 内部変数
      real*8, ALLOCATABLE ::buf(:)
      integer:: recv_status(MPI_STATUS_SIZE)! 受信ステータス
      integer:: n_member                 ! 担当部材数
      integer :: is                      ! 送信するデータの個数
      integer:: nn                       ! 確保するバッファのサイズ
c 本体
      n_member = Parameter_C.n_member
      nn = n_member*36
      ALLOCATE (buf(nn))
      is=0
      do i=1,n_member
      do j=1,18
      is=is+1
      buf(is)=Max_stress(i).stress_max_member(j)
      is=is+1
      buf(is)=Max_stress(i).stress_min_member(j)
      end do
      end do
      call MPI_Send(buf,is,MPI_DOUBLE,
     +              ID_MASTER,MPI_ANY_TAG,MPI_COMM_WORLD,ierr)
      DEALLOCATE (buf)
      return
      end
```

```
C-----------------------------------------------------------------------
C     ●   SUBROUTINE /recv_max_stress
C     ---------------------------
C     ●   各プロセスが計算した最大応力を取得
C-----------------------------------------------------------------------
      subroutine recv_max_stress( Parameter_C,Max_stress,
     +                                    nm_parallel,n_proc)
c 外部宣言
      use MPI_DEFINE
      implicit none
      include "..¥..¥sf3st¥submain.h"
C 引数宣言
      record /parameter_s    / Parameter_C
      record / max_stress_s  / Max_stress(*)
      integer:: n_proc
      integer:: nm_parallel(2, 0:n_proc)
C 内部変数
      real*8, ALLOCATABLE :: buf(:)
      integer:: n_member1, n_member2
      integer:: recv_status(MPI_STATUS_SIZE)
      integer:: nn
      integer:: k, kk, i, j, is
      integer:: ierr
c 本体
      nn = Parameter_C.n_member*36
      ALLOCATE (buf(nn))
      do k=1,n_proc-1
      call mpi_recv(buf, nn, mpi_double_precision,
     +              k, MPI_ANY_TAG, MPI_COMM_WORLD,
     +              recv_status,ierr)
c      kk = recv_status(MPI_STATUS_SOURCE)
      n_member1 = nm_parallel(1,k)    ! 担当部材の先頭番号
      n_member2 = nm_parallel(2,k)    ! 担当部材の最終番号
      is=0
      do i=n_member1,n_member2
      do j=1,18
      is=is+1
      Max_stress(i).stress_max_member(j)=buf(is)
      is=is+1
      Max_stress(i).stress_min_member(j)=buf(is)
      end do
      end do
      enddo
      DEALLOCATE (buf)
      return
       end
```

4.4.7 解析制御データ送受信

本節では、解析の流れを制御するコードの送受信について説明する。解析の流れの制御はマスター側で行い、その情報を全スレーブに送信して、スレーブ側もマスター側と全く同じ流れにするように制御する。

解析制御は２箇所で行い、次のコードで制御パラメータを送受信を行っている。最初が、マスター側の送信コード、次がスレーブ側の受信コードである。制御用パラメータの送受信コードが埋め込まれている２箇所とは、反復計算内の収束チェックを行った後と、次の計算ステップに移る前の最大変位をチェックを行った後である。いづれも、この部分でマスター側の解析フローに分岐点があり、どちらに進んだかを全スレーブに知らすために、この制御用パラメータの送受信が必要となる。

```
c──────────────────────────────────────★収束コードをスレーブに送る
c────── ★★スレーブに収束コードを転送
      if ( n_proc .ge. 2)then
c────── ★★  Time Check Codes
      call MPI_WTime(TIME_NET_CONV, nStart)
c──────────────────────────────────────
      call bcast_ite(ite_end)
```

```
c     write(damp_out,*) ' Cal_disp_vel ok'
c────── ★★収束したかチェック(mpp)    1:収束  0:未収束
c 同期
c     call synchronous_process( MPP_Ana_Group)
      call recv_ite(ite_end)
      if(ite_end.eq.1)go to 9980                        ! 収束していればループから抜ける
```

上記の２つのサブルーチンの内容を示す。

```
C──────────────────────────────────────────────────────
C     ●   SUBROUTINE /bcast_ite
C     ─────────────────
C     ●   反復計算終了コードを送る
C──────────────────────────────────────────────────────
      subroutine bcast_ite(ite_end)
c 外部宣言
      use MPI_DEFINE
      implicit none
c 引数
      integer:: ite_end          ! 反復計算終了コード(1:終了)
c 内部変数
      integer:: ierr
c 本体
      call MPI_Bcast(ite_end,1,MPI_INTEGER,ID_MASTER,
     +                              MPI_COMM_WORLD,ierr)
      return
      end
C──────────────────────────────────────────────────────
C     ●   SUBROUTINE /recv_ite
C     ─────────────────
C     ●   反復計算終了コードを受け取る
```

```
C-------------------------------------------------------------------
      subroutine recv_ite(ite_end)
c 外部宣言
      use MPI_DEFINE
      implicit none
c 引数
      integer:: ite_end          ! 反復計算終了コード(1:終了)
c 内部変数
      integer:: ierr
      integer:: recv_status(MPI_STATUS_SIZE) ! 受信ステータス
c 本体
      call MPI_Recv(ite_end,1,MPI_INTEGER,
     +    ID_MASTER, MPI_ANY_TAG,MPI_COMM_WORLD, recv_status, ierr)
      return
      end
```

分散並列型のシステムでは、極力データの送受信を減らすと共に、送受信する情報を統合化して、送信回数を減らす工夫が必要である。ここでは、この節で説明した 1 増分ステップ毎に 2 回の制御コード送信を取り除くこととする。この改良バージョンを Ver.1.11 とする。

改良バージョンVer.1.11について、変更点を説明する

この改良バージョンでは、マスターの反復過程でスレーブに送信する加速度ベクトルの最終項に、この 2 つの制御コードを含ませることにした。ここで使用する制御コードは S_control_code であり、以下の仕様である。

```
c------- ★★スレーブに解析開始コードを転送
c          スレーブ制御コード 0:  収束
c                              1:  未収束
c                              2:  構造物崩壊もしくはエラーは発生、途中終了
```

マスター反復過程、並びに最大変位チェックのコードを以下に示す。

```
c-------------------------------------------------★収束したかチェック(ok)
      ite_end=ICheck_error(iroop,n_point,Point,n_unknown,
     *         result_disp_point,est_disp_point, Newmark_P)
      if(S_control_code.ne.2) S_control_code = ite_end
c-------------------------------------------------★加速度をスレーブに送信(ok)
c------- ★★スレーブに計算結果を転送
      if( n_proc .ge. 2)then
c------- ★★  Time Check Codes 予測加速度の通信
      call MPI_WTime(TIME_NET_ACC, nStart)
c-----------------------------------------------------
       call Send_Results_acc(n_unknown,result_acc_point,
     *         nm_parallel,n_proc, Ms_table,Ms_free,
     *         S_control_code,buf_disp)
c------- ★★  Time Check Codes 予測加速度の通信
      call MPI_WTime(TIME_NET_ACC, dTime_NET_ACC)
      All_Time_NET_ACC = All_Time_NET_ACC + dTime_NET_ACC
                                                  ! 予測加速度通信
```

収束したかどうかチェックする

全ステップで、変位が最大値を超えた場合は、そのままの制御コードとする

加速度と共に制御コードを全スレーブに送信する

```
c-------------------------------------------------------
      endif
      if(S_control_code .eq. 2) goto 9998                    ! 解発散、後処理へ
      if(S_control_code .eq. 0) goto 9980                    ! 収束、応力計算へ
       .
       .
c-------------------------------------------------★終了か？（ステップと最大値チェック）
      if(istep .ge. Newmark_P.nn_step .or.
     *    d_max_v .gt.Control.collapse_maxdisp ) S_control_code=2
       .
       .
```

マスター側の制御は、ここで行う

最大値チェックで崩壊と見做された場合は、制御コードを２とする

次に、制御コードを受けるスレーブ側のコードを見よう。

```
c-------------------------------------------------★サーバーで計算した加速度を受け取る（vpp）
c-------- ★★マスターから計算結果加速度を取得
      call recv_result_acc(n_unknown,result_acc_point,ierr,
     *                     S_control_code)
c-------------------------------------------------★β法に基づき加速度より変位・速度を計算(ok)
       call Cal_disp_vel(n_unknown,
     *                result_disp_point, result_vel_point, result_acc_point,
     *                past_disp_point, past_vel_point, past_acc_point,
     *                Newmark_P)
c                                                  write(damp_out,*) ' Cal_disp_vel ok'
c-------- ★★収束したかチェック(mpp)    1:収束  0:未収束
c          スレーブ制御コード 0:  収束
c                           1:  未収束
c                           2:  発散・途中終了
c 同期
c     call synchronous_process( MPP_Ana_Group)
c     call recv_ite(ite_end)
      if(S_control_code.eq.0)go to 9980        ! 収束していればループから抜ける
      if(S_control_code.eq.2)go to 9998
c     write(damp_out,*) ' Check_error ok'
```

加速度と共に制御コードを受け取る

スレーブ側の制御は、ここで行う

次に、制御コードを含んだ加速度ベクトルをスレーブに送信するサブルーチンと受信するサブルーチンを以下に示す。

```
C-------------------------------------------------------------------
C     ●   SUBROUTINE /Send_Results_acc(マスター用)
C     ------------------------------------
C     ●   マスターが計算した加速度を圧縮し、スレーブに転送
C-------------------------------------------------------------------
      subroutine Send_Results_acc(n_unknown,result_acc_point,
     *         nm_parallel, n_proc, Ms_table, Ms_free,
     *         S_control_code,disp_S)
c   外部宣言
      use MPI_DEFINE
      implicit none
c     引数宣言
      integer :: n_unknown            ! マトリクスの未知数の数
```

```
      real*8  :: result_acc_point(*)  ! スレーブに送信する加速度
      integer :: n_proc               ! 総プロセス数
      integer :: nm_parallel(2, n_proc)   ! 部材担当範囲
      integer :: Ms_table(n_unknown, *)   ! データ変換用テーブル
      integer :: Ms_free(*)               ! 各スレーブの自由度
      integer :: S_control_code           ! スレーブコントロールコード
c     内部変数
c      real*8, ALLOCATABLE ::disp_S(:)    ! 変換後の加速度
      real*8 disp_S(*)
      integer :: k,i                      ! カウンタ
      integer :: ierr,msx
c     実装
c      ALLOCATE (disp_S(n_unknown+1))
      do k = 1, n_proc-1
      call Dt_from_M_to_S(disp_S,result_acc_point,
     *                    Ms_free(k),Ms_table(1,k))
      msx=Ms_free(k)+1
      disp_S(msx)=S_control_code+0.5
      call MPI_Send(disp_S, msx, MPI_DOUBLE, k,
     +              MPI_ANY_TAG, MPI_COMM_WORLD, ierr)
      enddo
c      DEALLOCATE (disp_S)
      return
      end
```

```
C-----------------------------------------------------------------------
C     ●   SUBROUTINE /recv_result_acc(スレーブ用)
C     ―――――――――――――――――――――
C     ●   計算加速度をマスターから受ける
C-----------------------------------------------------------------------
      subroutine recv_result_acc(n_unknown,result_acc_point,ierr,
     *                           S_control_code)
c     外部宣言
      use MPI_DEFINE
      implicit none
C     引数宣言
      Integer :: n_unknown                ! 未知数
      real*8  ::result_acc_point(*)       ! 加速度ベクトル
      integer :: ierr,S_control_code
C     内部変数
      Integer :: recv_status(MPI_STATUS_SIZE) ! 受信ステータス
      integer :: nx
c     実装
      nx=n_unknown+1
      call MPI_Recv(result_acc_point, nx, MPI_DOUBLE,
     +             ID_MASTER, MPI_ANY_TAG,MPI_COMM_WORLD, recv_status, ierr)
      S_control_code=result_acc_point(nx)
      return
      end
```

本節では、並列処理と単一処理との解析結果を比較し、並列処理で行った解析の妥当性をチェックする。

4.5 分散並列型動的解析結果の評価
4.5.1 ワーレントラス型アーチ

最初は、ステップ荷重を受けるワーレントラス型アーチの動的解析である。対象構造物は幾何学的非線形を考慮した平面構造であり、減衰はゼロとしている。節点変位は、(u,w)の２自由度である。解析では、幾何学非線形性を含んでいるため、アーチとしての動座屈、あるいは、部材の弾性座屈を考慮している。

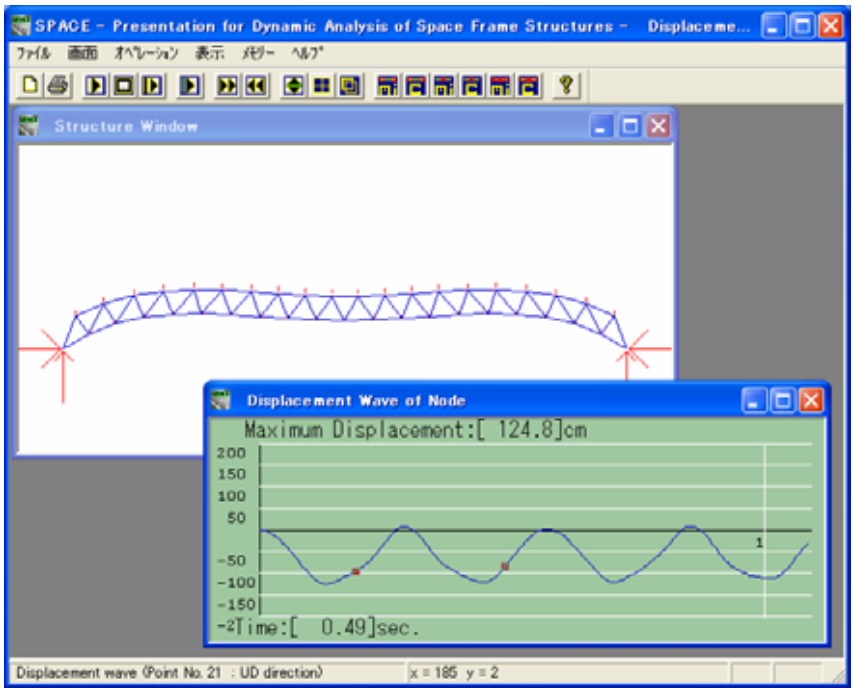

図 4-2 単一 CPU によるワーレントラス型アーチの解析結果

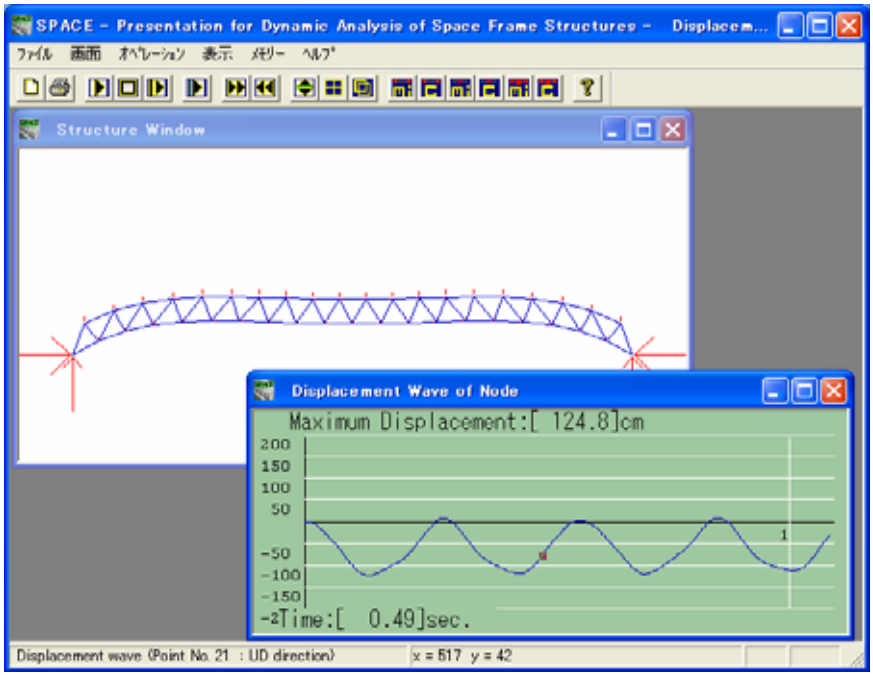

図 4-3 並列処理によるワーレントラス型アーチの解析結果

図 4-2 と 4-3 は、単一の CPU で計算した場合と並列処理で計算した結果を示すものである。中央の変位から分かるように全く同一の結果が得られている。

4.5.2 平面フレーム

ここでは、平面フレームについて解析し、単一の CPU で計算した結果と並列処理で計算した結果を比較する。両端ファイバーモデルを使用して弾塑性解析を行い、その結果を図 4-4 と 4-5 に示す。節点変位の時刻歴やフレーム変形状態、塑性ヒンジ発生状況、全て同じ結果が得られている。

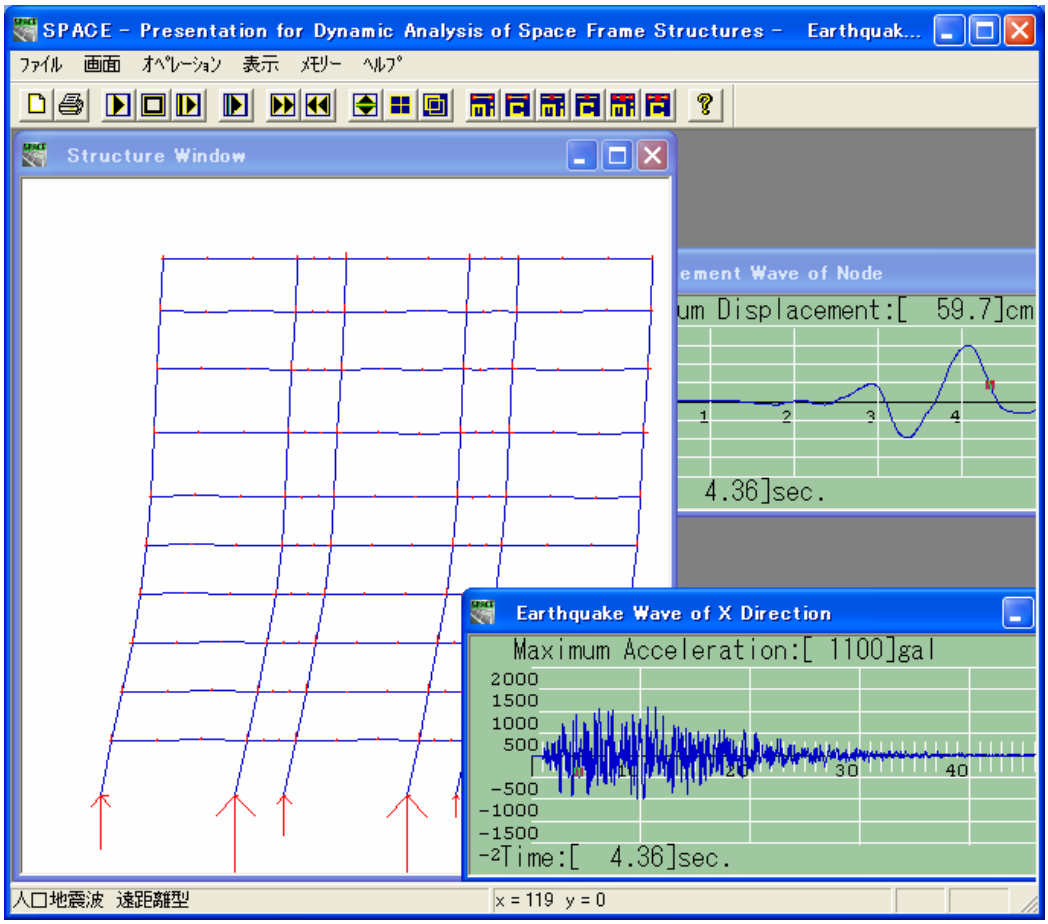

図 4-4 単一 CPU による平面フレームの弾塑性解析結果

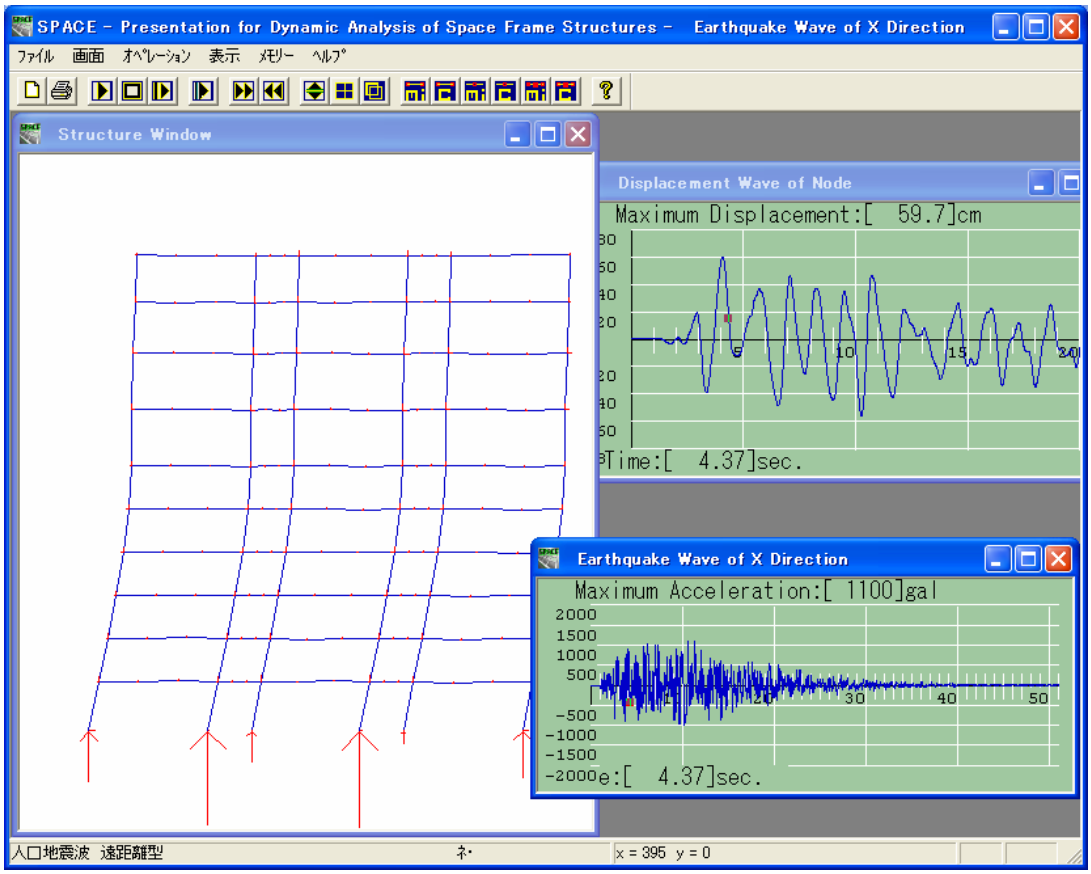

図 4-5 並列処理による平面フレームの弾塑性解析結果

本節では、分散並列型に拡張した各サブルーチンについて説明する。この拡張したサブルーチンは、前章で示したように次のようにまとめられている。また、サブルーチン名の右側の印＊は、前節で既に説明したことを示し、本節ではこれらのサブルーチンの説明は行わない。

１）分散並列型システム構築用

MPI_Comm_size()
MPI_Comm_rank()
RequestMember()

２）コントロールデータ送信のための情報収集

dyctl1_pa()+
dyctl1_set()+
dyctl2_pa()+
dyctl2_set()+
damctl_pa()+
damctl_set()+
doutcl_pa()+
doutcl_set()+
send_ctlset() +
recv_ctlset()+

３）データ送受信用

Send_structure()+
Send_imperfection()　＊
Send_Fiber()　＊
Send_RO_data() ＊
Send_damp()　＊
Convert_node_inf_vpp()　＊
Create_table()　＊
Reset_ij_table()　＊
recv_ld_repeat()　＊
send_ld_repeat()　＊
Send_Results_acc()　＊
send_force_vpp()　＊
bcast_ite()　＊
recv_force_vpp()　＊
recv_imperfection()　＊
recv_Fiber_P()　＊
recv_RO_data() ＊
recv_damp()　＊
recv_Results_acc()　＊
recv_ite()　＊
recv_max_stress() ＊
Dt_from_S_to_M()　＊
Dt_from_M_to_S()　＊

４）並列処理時間計算用

MPI_WTime()

５）解析用データ送信のための情報収集

Get_structure_pa()+
set_structure()+

```
          Get_imperfection_pa()   *
          recv_imperfection.for   *
          Fiber_input_pa()+
          recv_Fiber()+
          RO_data_input_pa()   *
```

６）既存ソフトの変更

```
          Get_pointforce_ld_pa()+
          Add_damp2_ld_pa()+
          Add_damp3_ld_pa()+
          Add_earth2_ld_pa()+
          Add_stiff1_ld_pa()+
          Add_stiff2_ld_pa()+
          Add_tan_stiff_ld_pa()+
          Add_fdd_ld_pa()+
          Cal_stress_pa ()+
          Check_stress_pa()+
          Check_Maxwell_stress_pa()+
          Get_max_stress_pa()+
          Get_nonlinear_stiff_pa()+
          out_section_check_pa()+
```

上で分類した各サブルーチンの役割について説明しよう。1）は、マスター側が、並列処理を行うために、SPACE システムから分担する部材番号やランクなどを取得するサブルーチンであり、これらについては、次章で説明する。2）は、制御データやコントロール情報をスレーブに送信するために、データをバッファ領域にパックするサブルーチン類である。ここでのプログラムは、単一 CPU プログラムを変更して用いている。3）は、スレーブとの送受信用サブルーチンと送信データをパックするためのテーブル作成サブルーチンである。これらのサブルーチンは、次節以降で詳細に説明する。

4）は、処理時間を計測するサブルーチンである。5）は、解析用データ送信のための情報収集用サブルーチンであり、既存の SPACE サブルーチンにデータをパックするためのコードを追加して用いている。内容については、次章を参照されたい。6）は、マスター側が分担する部材について処理を行うように、既存のプログラムコードの一部を変更して用いている。その内容は、既存のプログラムとほとんど同じである。

4.6.2 分散並列型システム構築用サブルーチン

本節では、分散並列型システム構築用のサブルーチンについて説明する。ただし、既に説明したものについては省略する。

この time_module は、処理時間を計測するための変数、パラメータを設定する。この MODULE は、マスター用動的ソルバーの主サブルーチン

にインクルードされる。

```
C     ------------------------------------------------------------
C     ●   MODULE / time_module
C     ------------------------------------------------------------
C     ●   時間を計測するためのパラメータ,変数定義
C     ------------------------------------------------------------
      MODULE TIME_MODULE
      implicit none

      integer, parameter ::    TIME_PRE          = 1        ! 予備計算
      integer, parameter ::    TIME_K            = 2        ! 係数行列作成
      integer, parameter ::    TIME_NEWMARK      = 3        ! Newmarkβ法
      integer, parameter ::    TIME_LD           = 4        ! 右辺項作成
      integer, parameter ::    TIME_NONLINER     = 5        ! 非線形項作成
      integer, parameter ::    TIME_CAL          = 6        ! 振動方程式を解く
      integer, parameter ::    TIME_DISP_VEL     = 7        ! 変位、速度を計算
      integer, parameter ::    TIME_STRESS       = 8        ! 応力の計算
      integer, parameter ::    TIME_F_STRESS     = 9        ! ファイバー応力の計算
      integer, parameter ::    TIME_KT           = 10       ! 接線剛性の計算
      integer, parameter ::    TIME_NET_LD       = 11       ! 右辺項の通信
      integer, parameter ::    TIME_NET_ACC      = 12       ! 予測加速度の通信
      integer, parameter ::    TIME_NET_CONV     = 13       ! 収束コードの通信
      integer, parameter ::    TIME_NET_FORCE    = 14       ! 部材節点力の通信
      integer, parameter ::    TIME_STEP         = 15       ! ステップ毎の通信
      integer, parameter ::    TIME_ALL          = 16       ! 総解析時間

      real*8 ::    dTime_PRE                     ! 予備計算
      real*8 ::    dTime_K                       ! 係数行列作成
      real*8 ::    dTime_NEWMARK                 ! Newmarkβ法
      real*8 ::    dTime_LD                      ! 右辺項作成
      real*8 ::    dTime_NONLINER                ! 非線形項作成
      real*8 ::    dTime_CAL                     ! 振動方程式を解く
      real*8 ::    dTime_DISP_VEL                ! 変位、速度を計算
      real*8 ::    dTime_STRESS                  ! 応力の計算
      real*8 ::    dTime_F_STRESS                ! ファイバー応力の計算
      real*8 ::    dTime_KT                      ! 接線剛性の計算
      real*8 ::    dTime_NET_LD                  ! 右辺項の通信
      real*8 ::    dTime_NET_ACC                 ! 予測加速度の通信
      real*8 ::    dTime_NET_CONV                ! 収束コードの通信
      real*8 ::    dTime_NET_FORCE               ! 部材節点力の通信
      real*8 ::    dTime_STEP                    ! ステップ毎の通信

      real*8, save ::    All_Time_PRE            ! 予備計算
      real*8, save ::    All_Time_K              ! 係数行列作成
      real*8, save ::    All_Time_NEWMARK        ! Newmarkβ法
      real*8, save ::    All_Time_LD             ! 右辺項作成
      real*8, save ::    All_Time_NONLINER       ! 非線形項作成
      real*8, save ::    All_Time_CAL            ! 振動方程式を解く
      real*8, save ::    All_Time_DISP_VEL       ! 変位、速度を計算
      real*8, save ::    All_Time_STRESS         ! 応力の計算
      real*8, save ::    All_Time_F_STRESS       ! ファイバー応力の計算
      real*8, save ::    All_Time_KT             ! 接線剛性の計算
      real*8, save ::    All_Time_NET_LD         ! 右辺項の通信
```

```
      real*8, save ::     All_Time_NET_ACC          ! 予測加速度の通信
      real*8, save ::     All_Time_NET_CONV         ! 収束コードの通信
      real*8, save ::     All_Time_NET_FORCE        ! 部材節点力の通信
      real*8, save ::     All_Time_STEP             ! ステップ毎の通信
      real*8, save ::     All_Time_ALL              ! 総解析時間

      END MODULE
```

サブルーチン RequestMember()は、プロセスのランクを取得するために用いられ、主サブルーチンの最初でコールされる。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /RequestMember
C       ----------------------------------------------------------
C     ●   指定ランクの担当部材を取得
C     ------------------------------------------------------------
      subroutine RequestMember( nm_parallel, nRank)
c     引数
      dimension nm_parallel(2,0:*)          ! 担当部材番号を格納する配列
      integer*4 :: nRank                    ! 取得したいプロセスのランク
c     実装
      call GetRange(1, nRank, nm_parallel(1, nRank))
      call GetRange(2, nRank, nm_parallel(2, nRank))
      write(76, '(A, i4, A, i10, A, i10)') 'rank:', nRank, '担当部材:',
     &            nm_parallel(1, nRank), '-', nm_parallel(2, nRank)
      end subroutine
```

4.6.3 コントロールデータ送信のための情報収集サブルーチン

本節では、コントロールデータ（解析制御情報）をパックし、スレーブにその情報を送信するためのサブルーチン、また、その情報を受信するサブルーチンについて説明する。ここでは、次の 10 個のサブルーチンについて説明する。

```
dyctl1_pa.for
dyctl1_set.for
dyctl2_pa.for
dyctl2_set.for
damctl_pa.for
damctl_set.for
doutcl_pa.for
doutcl_set.f
send_ctlset.for
recv_ctlset.for
```

動的解析ダイアログその１の情報のパックと設定を行うサブルーチンが以下に示される。サブルーチン dyctl1_pa()では、ファイル入力時

で、情報をパックするバッファを利用して、データ入力を行っている。
ここでは、バッファの配列位置に注意する必要がある。

```
C     ------------------------------------------------------------------
C     ●    SUBROUTINE /dyctl1_pa（チェック OK）
C     ------------------------------------------------------------------
C     ●    動的制御ファイル No.1 を入力(ok)（マスター用）
C     ------------------------------------------------------------------
      subroutine dyctl1_pa(ierr,nindis,gindis,f1sec,fs,fl,ifp,ist,
     *                     JIKUZERO,G_JIKUZERO_ALPH,
     *                              fd,id,iis)
      implicit real*8(A-H,O-Z)
      character fcode*6
      dimension id(*),iis(2)
      real*4 fd(*)
      dimension fs(10,3),fl(10,3),ifp(10,3),ist(3)

      ierr=0
      ihan=0
      numb=1
      fcode='dcontl'
      call datset(ihan,fcode,id ,fd )
      if(ihan.ne.0) then
      ierr=1
      return
      endif
      ifd=fd(1)+0.2
      iis(2)=iis(2)+id(1)+1  !入力個数をセット
      iis(1)=iis(1)+ifd+1    !入力個数をセット
      write(76,'(a,4i4)') ' 個数 ',id(1),ifd,iis(2),iis(1)
      ii1=id(1)
      write(76,'(10i5)') (id(i),i=1,ii1)
      write(76,'(5f14.5)') (fd(i),i=1,ifd)

      JIKUZERO=id(33)
      G_JIKUZERO_ALPH=fd(64)
      nindis=id(2)
      gindis=fd(2)
      f1sec=fd(3)
      do i=1,10
      fs(i,1)=fd(i+3)
      fs(i,2)=fd(i+23)
      fs(i,3)=fd(i+43)
      fl(i,1)=fd(i+13)
      fl(i,2)=fd(i+33)
      fl(i,3)=fd(i+53)
      ifp(i,1)=id(i+2)
      ifp(i,2)=id(i+12)
      ifp(i,3)=id(i+22)
      end do
      do i=1,3
      ist(i)=0
      if(f1sec.ne.0.) then
```

```
      do j=1,10
      if(ifp(j,i) .ne. 0 ) ist(i)=1
      end do
      endif
      end do
      return
      end
```

受信した情報を用いて、動的解析ダイアログその１のデータを設定する。このサブルーチンの中でコメントアウトしたデータ入力の替わりに受信したバッファ領域のデータを用いて設定する。

```
C     ------------------------------------------------------------
C     ●    SUBROUTINE /dyctl1_set
C     ------------------------------------------------------------
C     ●    動的制御ファイル No.1 をセット(ok) （スレーブ用）
C     ------------------------------------------------------------
      subroutine dyctl1_set(ierr,nindis,gindis,f1sec,fs,fl,ifp,ist,
     *    JIKUZERO,G_JIKUZERO_ALPH,fd,id)
      implicit real*8(A-H,O-Z)
      character fcode*6
      dimension id(100)
      real*4 fd(100)
      dimension fs(10,3),fl(10,3),ifp(10,3),ist(3)

c     ierr=0
c     ihan=0
c     numb=1
c     fcode='dcontl'
c     call datset(ihan,fcode,id ,fd )
c     if(ihan.ne.0) then
c     ierr=1
c     return
c     endif
      JIKUZERO=id(33)
      G_JIKUZERO_ALPH=fd(64)
      nindis=id(2)
      gindis=fd(2)
      f1sec=fd(3)
      do i=1,10
      fs(i,1)=fd(i+3)
      fs(i,2)=fd(i+23)
      fs(i,3)=fd(i+43)
      fl(i,1)=fd(i+13)
      fl(i,2)=fd(i+33)
      fl(i,3)=fd(i+53)
      ifp(i,1)=id(i+2)
      ifp(i,2)=id(i+12)
      ifp(i,3)=id(i+22)
      end do
      do i=1,3
      ist(i)=0
      if(f1sec.ne.0.) then
```

```
      do j=1,10
      if(ifp(j,i) .ne. 0 ) ist(i)=1
      end do
      endif

      end do
      return
      end
```

動的解析ダイアログその２の情報のパックと設定を行うサブルーチンが以下に示される。サブルーチン dyctl2_pa()では、ファイル入力時で、情報をパックするバッファを利用して、データ入力を行っている。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /dyctl2_pa （チェック OK)
C       ----------------------------------------------------------
C     ●   動的制御ファイル No.2 を入力(ok)（マスター用）
C     ------------------------------------------------------------
      subroutine dyctl2_pa(ierr,NSTEP,t,DELT,IGRA,IBETA,BETA,GAMMA,
     *        XGAL,NTIME,DLAMST,load_memb_mass,dt_M_filter,IT_ANALYS,
     *                     fd,id,iis)
      IMPLICIT REAL*8(A-H,O-Z)
      common /sf01/jdfile,kdfile,iidat,lengf,ltitle,timex,fnfile,title
      character fnfile(100)*50,title*50,timex*20(100)
      integer*4 jdfile(100),kdfile(100),lengf(100)
      character fcode*6
      dimension id(*),iis(2)
      real*4 fd(*)
      dimension IGRA(3),XGAL(3),DBETA(4),dt_M_filter(4)
      data DBETA/0.,4.,6.,8./

      njishin=56
      ierr=0
      ihan=0
      numb=1
      fcode='dcalcl'
      call DATSET(ihan,fcode,ID,FD)
      if(ihan.ne.0) then
      ierr=1
      return
      endif

      ifd=fd(1)+0.2
      iis(2)=iis(2)+id(1)+1  !入力個数をセット
      iis(1)=iis(1)+ifd+1    !入力個数をセット
      write(76,'(a,4i4)') ' 個数 ',id(1),ifd,iis(2),iis(1)
      ii1=id(1)
      write(76,'(10i5)') (id(i),i=1,ii1)
      write(76,'(5f14.5)') (fd(i),i=1,ifd)

      DELT = fd(3)
      IF(DELT.LE.0.) delt=0.005
      t=fd(2)
```

```
      nstep=t/delt
      do i=1,3
      igra(i)=id(i+1)
c     地震波のチェック
      if(jdfile( njishin+i).ne.1) igra(i)=0
      xgal(i)=fd(i+3)
      enddo
      ibeta=id(5)
      gamma=fd(7)
      ntime=id(6)
      load_memb_mass=id(7)
      IT_ANALYS=id(8)
      DLAMST=fd(8)
      IF(IBETA.EQ.1) THEN
      beta=0.D0
      ELSE
      BETA=1.D0/DBETA(IBETA)
      ENDIF
      if(IBETA.eq.4) then
      BETA = 0.3025d0
      endif
c                 Maxwell 用フィルターデータセット
      dt_M_filter(1)=fd(9)
      dt_M_filter(2)=fd(10)
      dt_M_filter(3)=fd(11)
      dt_M_filter(4)=fd(12)
c     write(76,'(a,4f12.2)') ' dt mfilter ',(dt_M_filter(k),k=1,4)
      return
      end
```

受信した情報を用いて、動的解析ダイアログその２のデータを設定する。このサブルーチンの中でデータ入力の替わりに、受信したバッファ領域のデータを用いて設定する。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /dyctl2_set
C       ----------------------------------------------------------
C     ●   動的制御ファイル No.2 をセット(ok) （スレーブ用）
C     ------------------------------------------------------------
      subroutine dyctl2_set(ierr,NSTEP,t,DELT,IGRA,IBETA,BETA,GAMMA,
     *        XGAL,NTIME,DLAMST,load_memb_mass,dt_M_filter,IT_ANALYS,
     *        fd,id)
      IMPLICIT REAL*8(A-H,O-Z)
      common /sf01/jdfile,kdfile,iidat,lengf,ltitle,timex,fnfile,title
      character fnfile(100)*50,title*50,timex*20(100)
      integer*4 jdfile(100),kdfile(100),lengf(100)
      character fcode*6
      dimension id(100)
      real*4 fd(100)
      dimension IGRA(3),XGAL(3),DBETA(4),dt_M_filter(4)
      data DBETA/0.,4.,6.,8./

      njishin=56
```

```
      DELT = fd(3)
      IF(DELT.LE.0.) delt=0.005
      t=fd(2)
      nstep=t/delt
      do i=1,3
      igra(i)=id(i+1)
c     地震波のチェック
c     if(jdfile( njishin+i).ne.1) igra(i)=0
      xgal(i)=fd(i+3)
      enddo
      ibeta=id(5)
      gamma=fd(7)
      ntime=id(6)
      load_memb_mass=id(7)
      IT_ANALYS=id(8)
      DLAMST=fd(8)
      IF(IBETA.EQ.1) THEN
      beta=0.D0
      ELSE
      BETA=1.D0/DBETA(IBETA)
      ENDIF
      if(IBETA.eq.4) then
      BETA = 0.3025d0
      endif
c                 Maxwell 用フィルターデータセット
      dt_M_filter(1)=fd(9)
      dt_M_filter(2)=fd(10)
      dt_M_filter(3)=fd(11)
      dt_M_filter(4)=fd(12)
c     write(76,'(a,4f12.2)') ' dt mfilter ',(dt_M_filter(k),k=1,4)
      return
      end
```

減衰制御データの情報パックと設定を行うサブルーチンが以下に示される。サブルーチン damctl_pa()では、ファイル入力時で、情報をパックするバッファを利用して、データ入力を行っている。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /damctl_pa
C     ------------------------------------------------------------
C     ●   減衰制御ファイルを入力する(ok)(マスター用)
C     ------------------------------------------------------------
      subroutine damctl_pa(ierr,NMREAD,ITYDP,NDMP,NDMP2,NHH,HH,QHH,
     *                     fd,id,iis)
      IMPLICIT REAL*8(A-H,O-Z)
      common /sf01/jdfile,kdfile,iidat,lengf,ltitle,timex,fnfile,title
      character fnfile(100)*50,title*50,timex*20(100)
      integer*4 jdfile(100),kdfile(100),lengf(100)
      character fcode*6
      dimension id(*),iis(2)
      real*4 fd(*)
      dimension HH(100),QHH(100)
```

```
      nmdxx=42
      ierr=0
      ihan=0
      numb=1
      fcode='damp  '
      call datset(ihan ,fcode,id ,fd )
      if(ihan.ne.0) then
      ierr=1
      return
      endif
      ifd=fd(1)+0.2
      iis(2)=iis(2)+id(1)+1  !入力個数をセット
      iis(1)=iis(1)+ifd+1    !入力個数をセット
      write(76,'(a,4i4)') ' 個数 ',id(1),ifd,iis(2),iis(1)
      ii1=id(1)
      write(76,'(10i5)') (id(i),i=1,ii1)
      write(76,'(5f14.5)') (fd(i),i=1,ifd)
c     NMREAD=ID(2)
      nmread=1
      if(nmread.eq.1) then
      if(jdfile( nmdxx).ne.1) nmread=0
      if(jdfile( nmdxx+1).ne.1) nmread=0

      endif
      ITYDP=ID(3)
      NDMP=ID(5)
      NDMP2=ID(6)
      IF(NDMP.LE.0) NDMP=1
      IF(NDMP2.LE.0) NDMP2=2

      IF(ITYDP.EQ.1.OR.ITYDP.EQ.2)THEN
      NHH=1
      HH(1)=FD(2)
      QHH(1)=FD(7)
      hh(2)=0.
      qhh(2)=0.
      ELSEIF(ITYDP.EQ.3) THEN
      NHH=2
      HH(1)=FD(2)
      QHH(1)=FD(7)
      HH(2)=FD(3)
      QHH(2)=FD(8)
      ELSEIF(ITYDP.EQ.4) THEN
      NHH=ID(4)

      IF(NHH.LE.0) NHH=2
      DO I=1,NHH
      HH(i)=FD(I+1)
      QHH(i)=FD(I+6)
      enddo
      ENDIF
      RETURN
      END
```

受信した情報を用いて、減衰制御ファイルのデータを設定する。このサブルーチンの中でデータ入力の替わりに、受信したバッファ領域のデータを用いて設定する。

```
C     ------------------------------------------------------------
C     ●    SUBROUTINE /damctl_set
C       ----------------------------------------------------------
C     ●    減衰制御ファイルをセット(ok) (スレーブ用)
C     ------------------------------------------------------------
      subroutine damctl_set(ierr,NMREAD,ITYDP,NDMP,NDMP2,NHH,HH,QHH,
     *                      fd,id)
      IMPLICIT REAL*8(A-H,O-Z)
      common /sf01/jdfile,kdfile,iidat,lengf,ltitle,timex,fnfile,title
      character fnfile(100)*50,title*50,timex*20(100)
      integer*4 jdfile(100),kdfile(100),lengf(100)
      character fcode*6
      dimension id(100)
      real*4 fd(100)
      dimension HH(100),QHH(100)

       nmdxx=42
       NMREAD=ID(2)
      ITYDP=ID(3)
      NDMP=ID(5)
      NDMP2=ID(6)
      IF(NDMP.LE.0) NDMP=1
      IF(NDMP2.LE.0) NDMP2=2

      IF(ITYDP.EQ.1.OR.ITYDP.EQ.2)THEN
      NHH=1
      HH(1)=FD(2)
      QHH(1)=FD(7)
      hh(2)=0.
      qhh(2)=0.
      ELSEIF(ITYDP.EQ.3) THEN
      NHH=2
      HH(1)=FD(2)
      QHH(1)=FD(7)
      HH(2)=FD(3)
      QHH(2)=FD(8)
      ELSEIF(ITYDP.EQ.4) THEN
      NHH=ID(4)
      IF(NHH.LE.0) NHH=2
      DO I=1,NHH
      HH(i)=FD(I+1)
      QHH(i)=FD(I+6)
      enddo
      ENDIF
      RETURN
      END
```

ファイル出力定義の情報のパックと設定を行うサブルーチンが以下

に示される。サブルーチン doutcl_pa()では、ファイル入力時で、情報をパックするバッファを利用して、データ入力を行っている。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /doutcl_pa（チェック OK）
C       ----------------------------------------------------------
C     ●   ファイル出力定義ファイルを入力する(ok)（マスター用）
C     ------------------------------------------------------------
      subroutine doutcl_pa(ierr,IWSTP,SOUTSC,DMAXCK,No_section,
     *                     fd,id,iis)
      IMPLICIT REAL*8(A-H,O-Z)
      character fcode*6
      dimension id(*),No_section(10),iis(2)
      real*4 fd(*)

      ierr=0
      ihan=0
      numb=1
      fcode='doutcl'
      call datset(ihan,fcode,ID,FD)
      if(ihan.ne.0) then
      ierr=1
      return
      endif
      ifd=fd(1)+0.2
      iis(2)=iis(2)+id(1)+1  !入力個数をセット
      iis(1)=iis(1)+ifd+1    !入力個数をセット
      write(76,'(a,4i4)') ' 個数 ',id(1),ifd,iis(2),iis(1)
      ii1=id(1)
      write(76,'(10i5)') (id(i),i=1,ii1)
      write(76,'(5f14.5)') (fd(i),i=1,ifd)

      IWSTP=ID(2)
      SOUTSC=FD(2)
      DMAXCK=FD(3)
      iix=1
      do i=1,10
      if(id(i+2).ne.0) then
      iix=iix+1
      No_section(iix)=id(i+2)
      endif
      enddo
      No_section(1)=iix-1
      RETURN
      END
```

受信した情報を用いて、ファイル出力定義のデータを設定する。このサブルーチンの中でデータ入力の替わりに、受信したバッファ領域のデータを用いて設定する。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /doutcl_set（チェック OK）
```

```
C     ------------------------------------------------------------
C     ●    ファイル出力定義ファイルを入力する(ok)（マスター用）
C     ------------------------------------------------------------
      subroutine doutcl_set(ierr,IWSTP,SOUTSC,DMAXCK,No_section,
     *                      fd,id)
      IMPLICIT REAL*8(A-H,O-Z)
      character fcode*6
      dimension id(*),No_section(10)
      real*4 fd(*)

      IWSTP=ID(2)
      SOUTSC=FD(2)
      DMAXCK=FD(3)
      iix=1
      do i=1,10
      if(id(i+2).ne.0) then
      iix=iix+1
      No_section(iix)=id(i+2)
      endif
      enddo
      No_section(1)=iix-1
      RETURN
      END
```

このサブルーチン send_ctlset()では、解析制御情報をパックしたバッファ領域を全スレーブに送信する。このとき、整数バッファの最初の8つの領域に解析用コントロール情報、節点数、要素数、部材数などをセットする。最初の送信は、2つのデータ、実数データ数と整数データ数であり、次に送信するデータ数を表す。

```
C     ------------------------------------------------------------
C     ●    SUBROUTINE /send_ctlset
C     ------------------------------------------------------------
C     ●    制御情報をスレーブに転送する(ok)
C     ------------------------------------------------------------
      subroutine send_ctlset(Parameter_C,fd,id,iif_dt, N_analysis)
c     外部宣言
      use MPI_DEFINE
      IMPLICIT REAL*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
c     引数
      record /parameter_s    / Parameter_C   ! 構造基本データ
      real*4 fd(100)
      integer :: id(100), iif_dt(2)
c     実装
      id(1)=N_analysis
      id(2)=Parameter_C.n_point            !node
      id(3)=Parameter_C.n_element          !nelem
      id(4)=Parameter_C.n_member           !memb
      id(5)=Parameter_C.n_boundary_p       !nrbound
      id(6)=Parameter_C.n_local_coord      !locod
      id(7)=Parameter_C.n_rot_axis         !njiku
```

```
        id(8)=Parameter_C.n_free              !6
        write(76,'(a,2i5)') ' 制御データ',iif_dt(2),iif_dt(1)
        write(76,'(10i8)') (id(i),i=1,iif_dt(2))
        write(76,'(10f10.2)') (fd(i),i=1,iif_dt(1))
c------- ★★  制御情報と構造用制御情報を送信        ★★-------
        call mpi_bcast(iif_dt, 2, MPI_INTEGER,ID_MASTER,
     +                                        MPI_COMM_WORLD,ierr)
        call mpi_bcast(id,iif_dt(2),MPI_INTEGER,ID_MASTER,
     +                                        MPI_COMM_WORLD,ierr)
        call mpi_bcast(fd,iif_dt(1),MPI_REAL,ID_MASTER,
     +                                        MPI_COMM_WORLD,ierr)
c------- ★★  ------------------------------------ ★★-------
        return
        end
```

このサブルーチン recv_ctlset()は、マスターより送られてきた制御情報を受信する。受信した後、整数データの最初の８つを構造体にセットする。また、この受信した情報は、動的バッファ領域にパックされており、これを制御パラメータにセットするためにバッファ領域の先頭番地をセットする。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /recv_ctlset
C     ------------------------------------------------------------
C     ●   制御ファイルをマスターより受信(ok)（スレーブ用）
C     ------------------------------------------------------------
      subroutine recv_ctlset(Parameter_C,ff_data,if_data,iif_dt,
     *                       N_analysis,MPP_Ana_Group,ierr_dat)
c     外部宣言
      use MPI_DEFINE
      IMPLICIT REAL*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
c     引数宣言
      record /parameter_s    / Parameter_C
      integer      :: if_data(*), iif_dt(2,4)
      real*4 :: ff_data(*)
c     内部変数
      integer      :: ii_dt(2)                          ! 受信するデータの個数
c     本体
c------- ★★  制御情報と構造用制御情報を受信        ★★-------
      call MPI_Bcast(ii_dt, 2, MPI_INTEGER, ID_MASTER
     +                                    ID_MASTER, MPI_COMM_WORLD, ierr)
      write(76,'(a,2i4)') '  制御データ',ii_dt(2),ii_dt(1)
      call MPI_Bcast(if_data, ii_dt(2), MPI_INTEGER,
     +                                    ID_MASTER, MPI_COMM_WORLD, ierr)
      write(76,'(10i8)')(if_data(i),i=1,ii_dt(2))
      call MPI_Bcast(ff_data, ii_dt(1), MPI_REAL,
     +                                    ID_MASTER, MPI_COMM_WORLD, ierr)
      write(76,'(10f10.2)')(ff_data(i),i=1,ii_dt(1))
c------- ★★  ------------------------------------ ★★-------
      N_analysis          = if_data(1)
      Parameter_C.n_point = if_data(2)             !node
```

```
      Parameter_C.n_element = if_data(3)          !nelem
      Parameter_C.n_member  = if_data(4)          !memb
      Parameter_C.n_boundary_p  = if_data(5)      !nrbound
      Parameter_C.n_local_coord  = if_data(6)     !locod
      Parameter_C.n_rot_axis  = if_data(7)        !njiku
      Parameter_C.n_free  = if_data(8)            !6
      iif_dt(1,1)=1
      iif_dt(2,1)=9
c-----------------------------------------------★制御データの先頭番地をセット
      do i=1,3
      i1=iif_dt(1,i)
      i2=iif_dt(2,i)
      iif_dt(2,i+1)=iif_dt(2,i)+if_data(i2)+1
      iif_dt(1,i+1)=iif_dt(1,i)+ff_data(i1)+1
      write(76,'(a,2i4)') ' iif_dt ',iif_dt(1,i+1),iif_dt(2,i+1)
      enddo
      RETURN
      END
```

4.6.4 データ送受信用サブルーチン

本節では、データ送受信に関連するサブルーチンの説明を行う。このデータ送受信に関連するサブルーチンの多くは、既に説明してきた。残りのサブルーチンは、以下に示すSend_structure()である。このサブルーチンは、パックした構造データを全スレーブに転送する。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /Send_structure
C       ----------------------------------------------------------
C     ●   構造データをスレーブに転送する
C     ------------------------------------------------------------
      subroutine Send_structure(buf,ibuf,id_buf,jd_buf)
c     外部宣言
      use   MPI_DEFINE
      implicit real*8(A-H,O-Z)
c     引数の宣言
      integer:: ibuf(*)          ! 整数データバッファ
      real*8:: buf(*)            ! 実数データバッファ
      integer :: id_buf          ! 整数バッファの数
      integer :: jd_buf          ! 実数バッファの数
c     内部変数の宣言
      parameter(damp_out = 76)
c     実装
      write(76,'(a,2i4)') ' send_structure ',jd_buf,id_buf
      write(76,'(10i8)') (ibuf(i),i=1,jd_buf)
      write(76,'(10f10.2)') (buf(i),i=1,id_buf)
c------ ★★  制御情報と構造用制御情報を受信        ★★------
      call mpi_bcast(ibuf,jd_buf,MPI_INTEGER,
     +                    ID_MASTER,MPI_COMM_WORLD,ierr)
      call mpi_bcast(buf,id_buf,MPI_DOUBLE,
     +                    ID_MASTER,MPI_COMM_WORLD,ierr)
c------ ★★  ------------------------------------ ★★------
```

```
        return
        end
```

ここで説明したサブルーチン以外で、分散並列型として元のコードを変更したものについては、全て付録に掲載する。